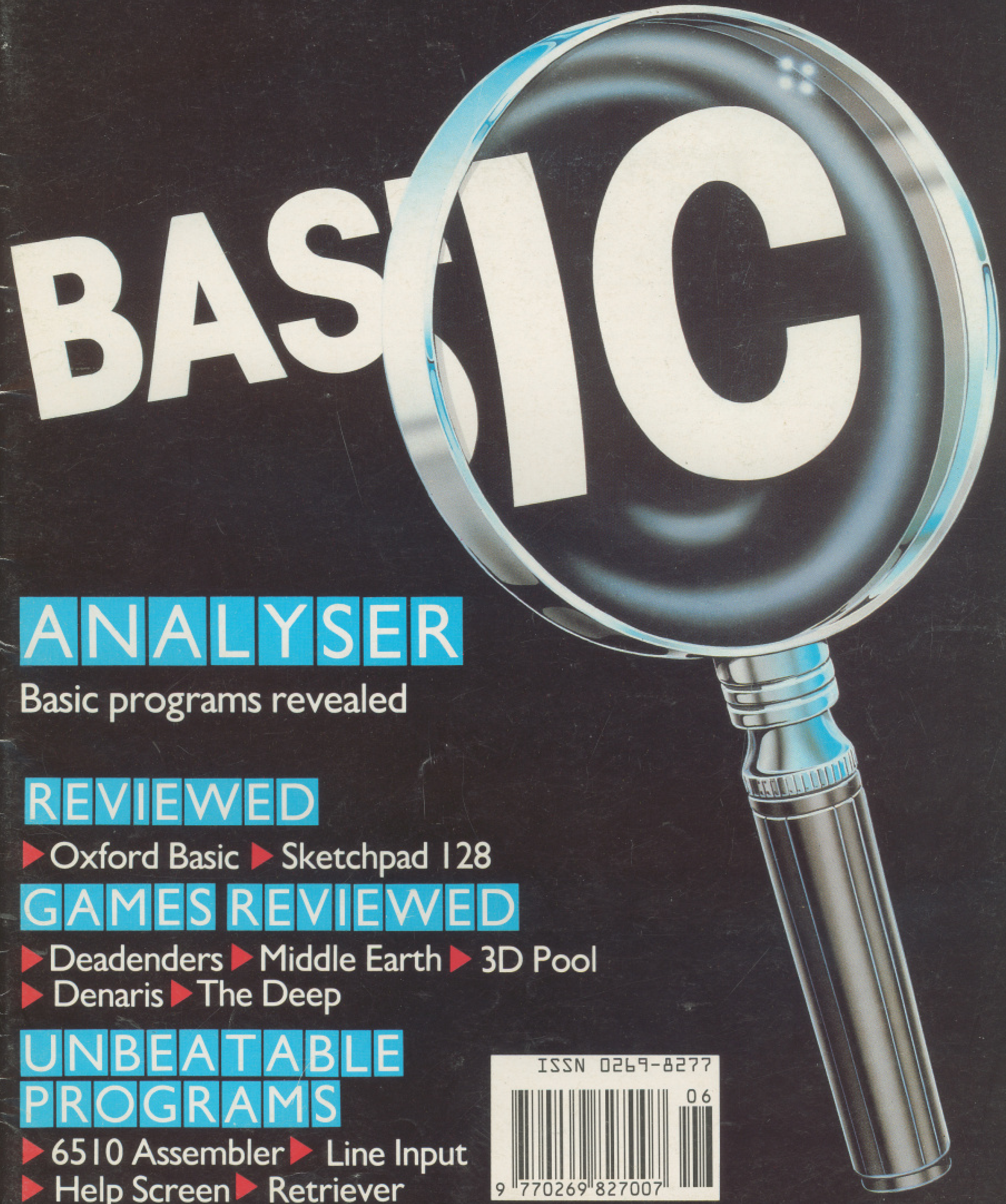
YOUR

COMMODORE

AN ARGUS SPECIALIST PUBLICATION

June 1989 £1.30



ANALYSER

Basic programs revealed

REVIEWED

▶ Oxford Basic ▶ Sketchpad 128

GAMES REVIEWED

▶ Deadenders ▶ Middle Earth ▶ 3D Pool
▶ Denaris ▶ The Deep

UNBEATABLE PROGRAMS

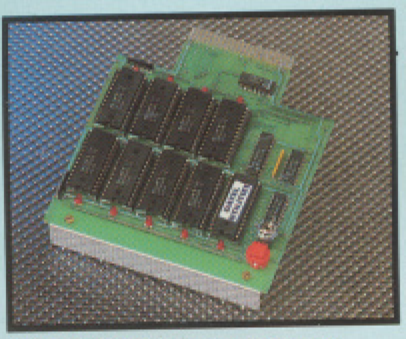
▶ 6510 Assembler ▶ Line Input
▶ Help Screen ▶ Retriever

ISSN 0269-8277

06

9 770269 827007

DATEL ELECTRONICS



256K SUPEROM EXPANDER

- Select instantly from 8 sockets which accept up to 32K EPROM each.
- Program your own EPROMs using our EPROM programer.
- No need to have lots of cartridges - just make a selection from the Superom menu.
- Fully menu driven on power up.
- Select any slot under software controls.
- Unique EPROM generator feature will take your own programs - basic or m/c & turn them into autostart EPROMs. (EPROM burner required).
- Accepts 2764/27128/27256 EPROMs.
- On board unstoppable reset.
- On board operating systems - no programs to load.

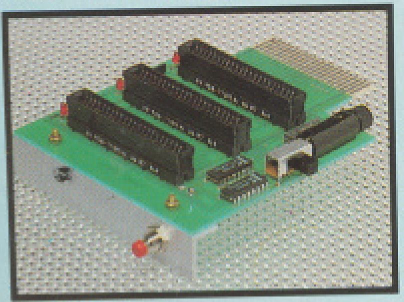
ONLY £29.99



RESET CARTRIDGE

- Unstoppable reset button. NOTE:- user port cheaper type reset buttons offered by others are not unstoppable.
- Resets even so called "unstoppable" programs.
- Add pokes from magazines etc.
- Fits in cartridge port.
- Simply plug in.

ONLY £5.99

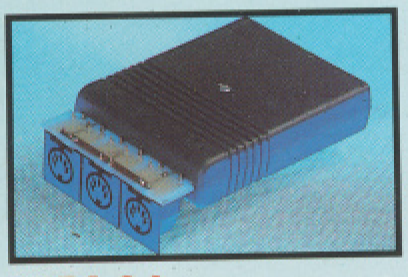


3 SLOT MOTHERBOARD

SAVE WEAR & TEAR ON YOUR EXPANSION PORT

- Accepts 3 cartridges.
- Switch in/out any slot.
- Onboard safety fuse.
- Fully buffered.
- High grade PCB.
- Reset button.

ONLY £16.99



MIDI 64

FULL MIDI INTERFACE FOR THE 64/128 AT A REALISTIC PRICE.

- MIDI in/MIDI thru/2 x MIDI out.
- Compatible with most leading software packages.

ONLY £29.99



DIGITAL SOUND SAMPLER

- The new sampler allows you to record any sound digitally into memory & then replay it with astounding effects.
- Playback forwards/backwards with echo/reverb/ring modulation.
- Now with full sound editing module to produce outstanding effects.
- Full 8 bit D to A & ADC conversion.
- MIDI compatible with suitable interface. (i.e. Datel unit for £29.99, see ad.).
- Live effects menu includes real time display of waveforms.
- Line in/mic in/line out/feedback controls.
- Powerfull sequencer with editing features.
- Load/save sample.
- Up to 8 samples in memory at one time.
- Tape or disk (please state).
- Complete software/hardware package.

ONLY £49.99

COM-DRUM

DIGITAL DRUM SYSTEM

- Now you can turn your digital sound sampler into a digital drum system.
- 8 digital drum sounds in memory at one time.
- Complete with 3 drum kits.
- Real drum sounds - not synthesised.
- Create superb drum rhythms with real & step time.
- Full editing. Menu driven.
- Output to hi-fi or through TV speaker.
- Load/save facilities.

ONLY £9.99

STATE TAPE OR DISK

COM-DRUM EDITOR

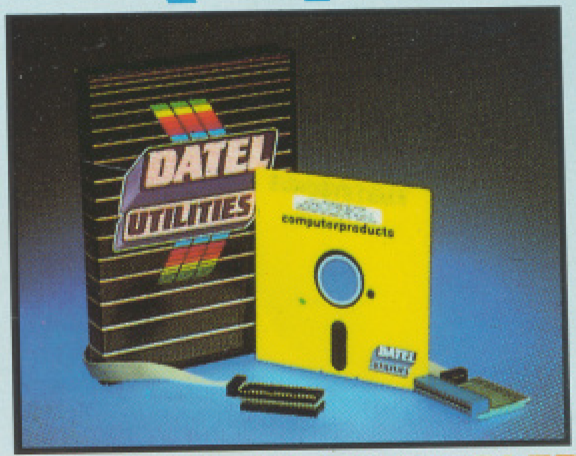
- 24 drum sounds supplied on disk to enable you to construct your own drum kit.
- Re-arrange sounds supplied with a Com-Drum to make new kits.
- Load/save facilities.

ONLY £4.99 DISK ONLY

DATA RECORDER

- Quality Commodore compatible data recorder.
- Pause control, counter, etc.
- Suitable for 64/128.
- Send now for quick delivery.

ONLY £24.99



DEEP SCAN BURST NIBBLER™

- The most powerfull disk nibbler available anywhere, at any price! Burst Nibbler is actually a two part system - a software package & a parallel cable to connect the 1541/1570/1571 to 64/128 (state type).
- What gives Burst Nibbler it's power? Conventional nibblers have to decode the data from the disk before it can transfer it using the serial bus - when non standard data is encountered they are beat. Burst Nibbler transfers data as raw GCR code via the parallel cable without the need to decode it so you get a perfect copy of the original.
- • Will nibble up to 41 tracks • Copy a whole disk in under 2 minutes • Full instructions • Regular updates - we always ship the latest • Fitted in minutes - no soldering usually required • Full or 1/2 tracks • No need to buy parallel cable if you have Disk Demon/Dolphin etc. • Cable has throughbus extension for other add ons.
- Wether to choose FastHack'em or Burst Nibbler? Fast Hack'em is unbeatable value as an "all rounder" - with nibblers, 1 or 2 drive copy, format, file copy, 1571 copy etc. etc., so if you have a more general requirement perhaps Fast Hack'em is for you. Burst Nibbler is a pure nibbler second to none, for the reasons stated. So if it's just making backups you are interested in, there is no other product to beat it!

ONLY £24.99 COMPLETE

SOFTWARE ONLY £12.99

CABLE ONLY £14.99

NOTICE 1988 COPYRIGHT ACT

DATEL ELECTRONICS Ltd. neither authorizes or condones the use of it's products to reproduce copyright material. It is illegal to make copies of such material without the expressed consent of the copyright owners or thier licencees.



ROBOTEK 64

MODEL & ROBOT CONTROL MADE EASY.

- 4 output channels - each with onboard relay.
- 4 input channels - each fully buffered TTL level sensing.
- Analogue input with full 8 bit conversion.
- Voice input for voice control.
- Software features:- test mode/analogue measurement/voice activate/digital readout etc.

ONLY £39.99

INCLUDING HARDWARE/SOFTWARE/MIC. ETC,(STATE TAPE OR DISK).

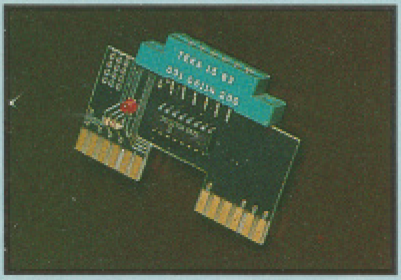
TOOLKIT IV

THE ULTIMATE DISK TOOLKIT FOR THE 1540/1541.

- A disk toolkit is an absolute must for the serious disk hacker. Toolkit IV has more features than most for less.
- DISC DOCTOR V2 - Read & write any track & sector including extra & renumbered tracks. Repair damaged sectors. Look underneath read errors.
- HEADER/GAP EDITOR - Decodes & displays ALL header information including off bytes & header gap. Rewrite the entire header & header gap. Renumber sectors. Also edit any sector tail gap.
- DISK LOOK - Sort directory. Recover lost files. Display file start /end addresses. Disassemble any file program directly from the disk to SCREEN or PRINTER including undocumented opcodes. Edit Bam.
- FAST FILE COPY - Selective file copy. Works at up to 6 times normal speed.
- FAST DISK COPY - Copy an entire disk in 2 minutes or less using single 1541.
- FILE COMPACTOR - Can compact machine programs by up to 50%. Save disk space. Compacted programs run as normal.
- FORMATTER - 10 second format an entire disk or format any individual track or half track 0 to 41. Re-define any of 30 parameters to create or recreate unique disk formats.
- ERROR EDIT - Quickly find & recreate all read errors including extras & renumbered tracks or sectors & half tracks from 0 to 41. Even recreates data under errors & allows you to redefine any necessary parameters.

ONLY £9.99

DATEL ELECTRONICS



NEW CLONEMASTER

- The most effective tape to tape back-up board available.
- Makes perfect backups of your tapes easily & effectively!!
- No user knowledge needed at all.
- On board TTL logic circuitry actually shapes the program & sends a perfect signal to the record cassette - producing a copy better than the original in many cases.
- L.E.D. indicator shows when data is being transferred to avoid exessive tape winding.
- Works with almost any program including multi-loaders, turbos & even very unusual type turbos.
- Requires access to two CBM compatible data recorders.
- Simply press 'Play' on one recorder & press 'Record' on the other - that's it!
- You can even make a backup while you are loading the program.
- This is a total hardware solution - no programs to load - the results are stunning!

ONLY £9.99

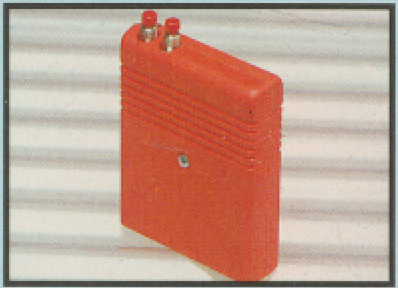


TURBO ROM II

TURBO ROM II IS A REPLACEMENT FOR THE ACTUAL KERNAL INSIDE YOUR 64. IT PROVIDES SUPERFAST LOAD/SAVE ROUTINES.

- Loads most programs at 5-6 times normal speed.
- Saves at 5-6 times normal speed as well.
- Improved DOS support including 10 sec. format.
- Programed function keys:- load, directory, old, etc.
- Return to normal kernal at a flick of a switch.
- FCOPY-250 block file copier.
- FLOAD-special I/O loader.
- Plus lots more.
- Fitted in minutes - no soldering usuallly required. (On some 64's the old ROM may have to be desoldered).

ONLY £14.99



SMART CART

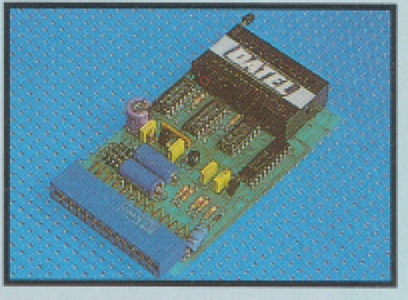
- 32K pseudo ROM.
- Lithium battery lasts up to 5 years.
- Simply load the program you require - then flick the switch. The cartridge can then be removed just like a ROM cartridge.
- Make your own cartridges including autostart types - without EPROM burner. 32K version = 4 x 8K pages.
- Some knowledge of M/C is helpful - but full instructions included.
- I/O 2 slot open for special programing techniques.

ONLY £29.99

RAM DISK

- Turn your Smart Cart into a 32K Ram/disk.
- 32K of instant storage area for files/programs.
- Disk type commands:- load, save, directory, scratch.
- Program data retained when computer is switched off!
- Full command set with instructions.

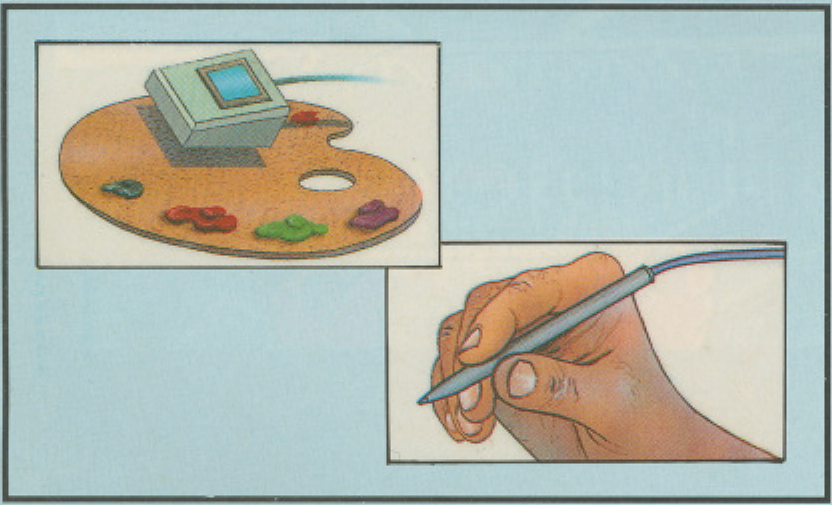
ONLY £9.99



EPROMMER 64™

- A top quality, easy-to-use EPROM programer for the 64/128.
- Fully menu driven software/ hardware package makes programing/reading/verifying/ copying EPROMs simplicity itself.
- Will program 2716, 2764, 27128 & 27256 chips, 12.5, 21 or 25 volts.
- Fits into user port for maximum compatibility with cartridges/ Superom Board etc.
- Full feature system - all functions covered like device check/verify.
- We believe Eprommer 64 is the most comprehensive, most friendly & best value for money programer available for the 64/128.
- Ideal companion for Superom Board, Cartridge Development System, our kernal expanders or indeed any EPROM base project.
- Comes complete with instructions - plus the cartridge handbook.

ONLY £39.99 COMPLETE



BLAZING PADDLES

A COMPLETE LIGHTPEN/GRAPHICS ILLUSTRATOR PACKAGE.

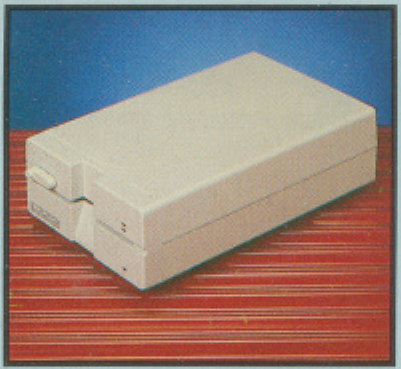
- A fully Icon driven graphics package of a calibre which should cost much, much more.
- Complete with a fibre optical lightpen system for pin point accuracy.
- Pictures can be saved from 'Action Replay' & edited with Blazing Paddles.
- Blazing Paddles will also work with many other input devices including:- Joysticks, Mice, Graphics Tablets, Trackball, etc.
- Multi feature software including:-
  - Range of brushes ● Airbrush
  - Rectangle ● Circle
  - Rubberbanding ● Lines
  - Freehand ● Zoom mode
  - Printer dump ● Load/save
  - Advanced colour mixing - over 200 hues! ● Cut & paste allows shapes/windows/pictures to be saved to/from tape/disk.

ONLY £24.99

COMPLETE BLAZING PADDLES & LIGHTPEN

BLAZING PADDLES ONLY £12.99

BLUE CHIP DRIVE



- External power supply for cool operation.
- Even works in 1571 double sided mode (128 mode).
- Fully C64/128 compatible.
- This drive is now probably the most compatible drive available for the Commodore. More so than even Commodores own '1541C'.
- Supplied complete with all cables - no more to buy.
- At last, a top quality drive at a sensible price.

ONLY £129.99

ADD £5.00 FOR COURIER DELIVERY

ULTRA CRUNCHER

- The ultimate cartridge based program compactor.
- Compacts by up to 30%!
- 3 compacting programs on one cartridge.
- Fast loading/saving routines.
- Full DOS support including fast format.

ONLY £12.99 on cartridge

Available as chip only for 256K Superom Board fcr ONLY £7.99

FAST HACK'EM™

- Multi Module Disk Nibbler - all on the one disk.
- Single 1541 Nibbler - Copy an entire disk in just 2 minutes.
- Super fast File Copy - typically copies an average file in under 9 seconds.
- No other product is such a good "All Rounder" at this incredibly low price...

ONLY £9.99

ALL ORDERS NORMALLY DESPATCHED WITHIN 48 HRS

HOW TO ORDER ...

BY PHONE	BY POST	FAX
0782 744707 24hr Credit Card Line	Send cheques/POs made payable to "Datel Electronics"	0782 744292 UK ORDERS POST FREE EUROPE ADD £1 OVERSEAS ADD £3

PRICES AND SPECIFICATIONS CORRECT AT TIME OF PRESS AND SUBJECT TO CHANGE WITHOUT NOTICE

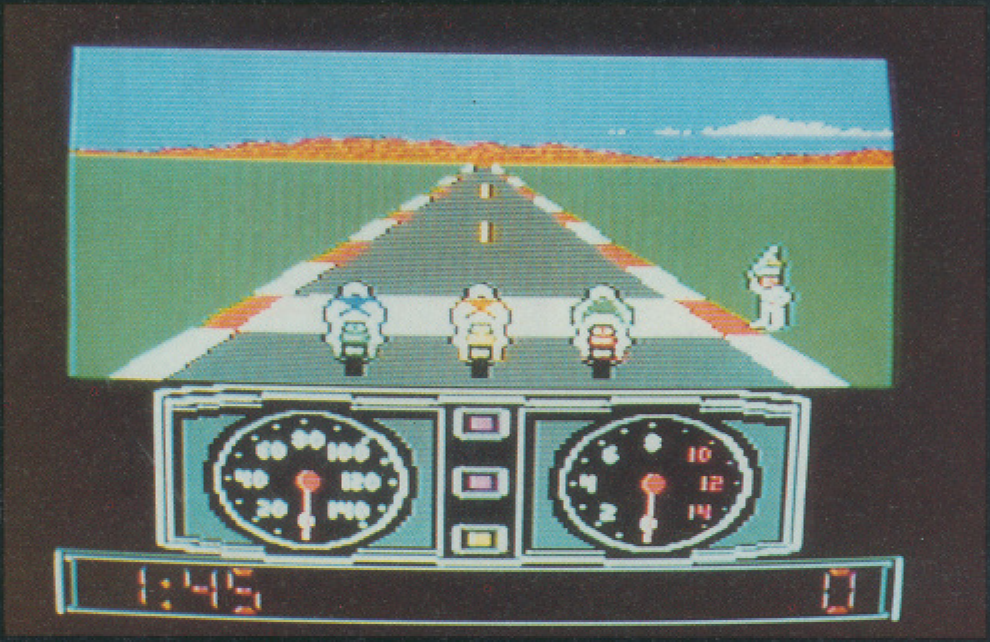
CALLERS WELCOME - Please reserve goods by telephone prior to visit.

DATEL ELECTRONICS

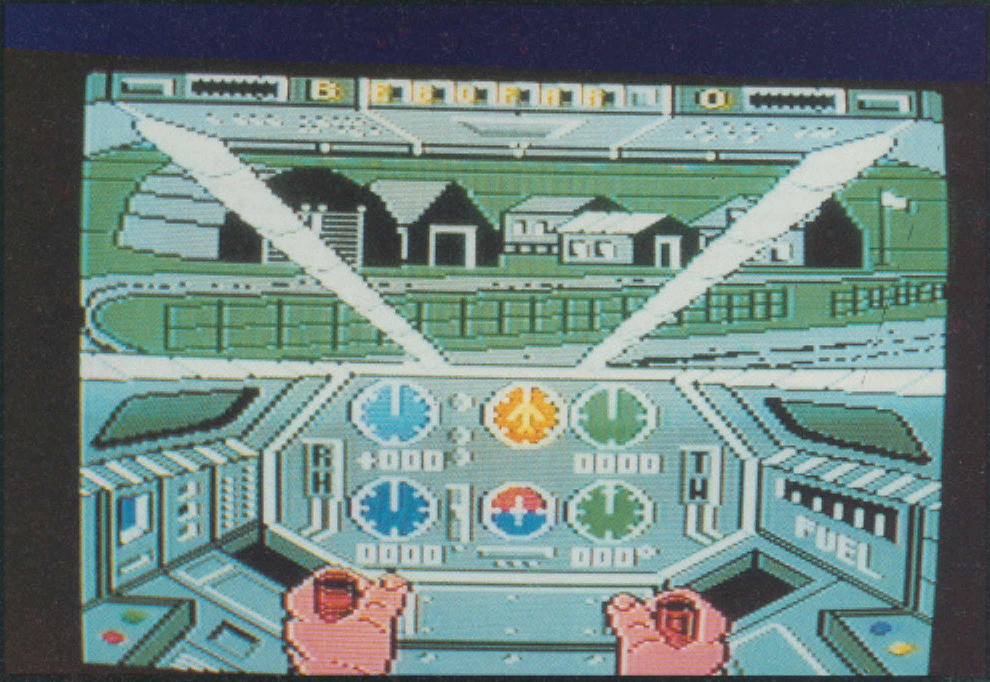
DATEL ELECTRONICS LTD., FENTON INDUSTRIAL ESTATE GOVAN ROAD, FENTON, STOKE-ON-TRENT, ENGLAND.

SALES ONLY	TECHNICAL ONLY
0782 744707	0782 744324

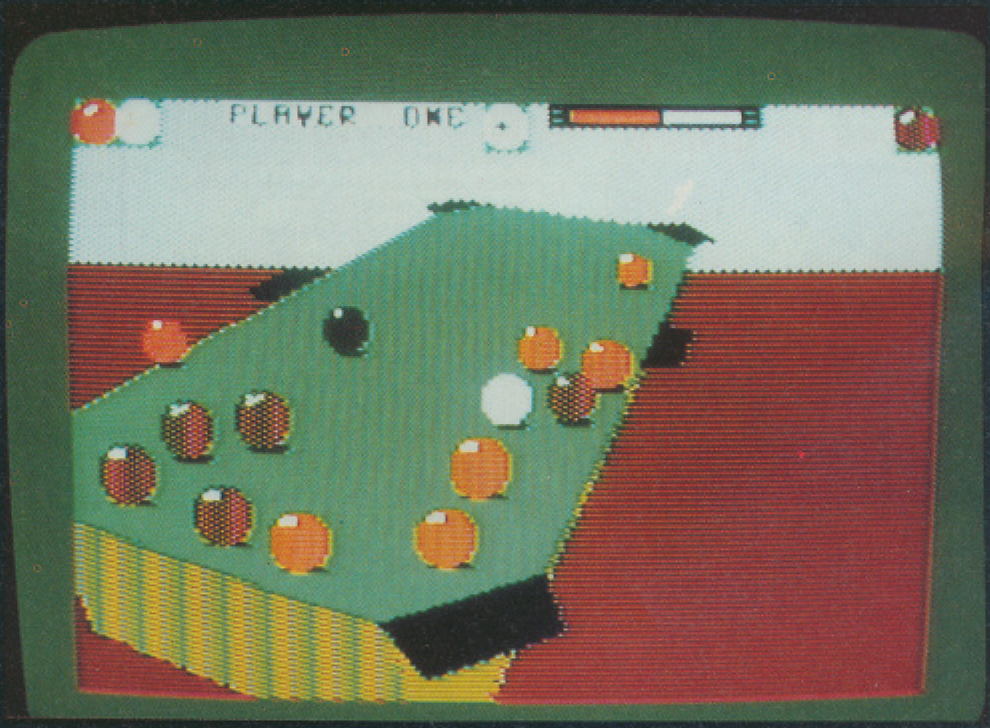
CONTENTS



Super Cycle



Infiltrator



3D Pool

**VOLUME 5
NUMBER 9**

FEATURES

REGULARS

PROGRAMS

Editor: *Stuart Cooke*
Deputy Editor:
Paul Whitington
Technical Editor: *Paul Eves*
Group Editor:
Mark Webb
Advertisement Manager:
Paul Kavanagh
Ad-Copy Control:
Ruth Taylor
Artist: *Alan Batchelor*
Designer: *Neil Sweetman*
Origination: *Ebony*
Printers: *Chase Webb*

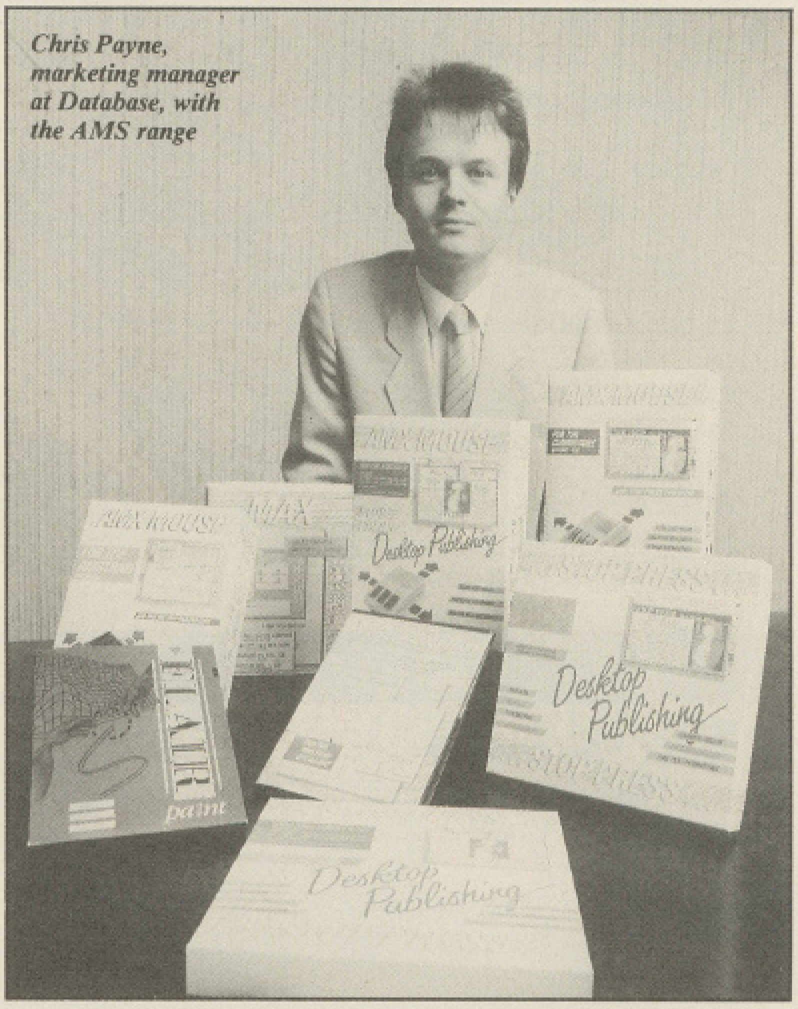
Your Commodore incorporating Your 64 is a monthly magazine appearing on the first Friday of each month. Argus Specialist Publications Limited. Editorial and Advertisement office, *Your Commodore,* Argus House, Boundary Way, Hemel Hempstead HP2 7ST. Telephone: (0442) 66551. Subscription rates upon application to Your Commodore Subscriptions Department, Infonet Ltd, 5 River Park Estate, Berkhamsted, Herts HP4 1HL. U.S.A. Subscrpition Agent: Wise Owl Worldwide Publications, 4314 West 238th Street, Torrance CA 90505 U.S.A.

ARGUS PRESS GROUP

ISSN 0269-8277

ABC

Data Statements



Chris Payne, marketing manager at Database, with the AMS range

Free The Spirit

If there's one company above all the others that's trying to make your C128 more like an Amiga, it's Free Spirit Software. This American company seems determined to produce good, high-quality graphics packages for the C128. Their latest offering is Poster Maker 128, with which you can create posters ranging in size up to 25 pages. You need 80 col display and a C128D or C128 with the 64K video Ram upgrade. You can import Basic 8, Sketchpad 128 or Spectrum 128 files. There is also an option for reducing, thereby enabling you to perform clip art.

***Touchline:** Free Spirit Software, PO Box 128, 58 Noble St, Kutztown, PA 19530 **Tel:** 215 683 5609.*

Spririt of Adventure

Good news for Adventurers. After much public pressure, Incentive Software has rewritten the classic Graphic Adventure Creator. The new release will allow 'Stand alone disk accessed' adventures to be produced. This will mean that you can now create adventures of incredible size and complexity, restricted only by disk size. The release date for this exciting product is April 10th. It will retail at £29.95, but existing GAC holders may upgrade for a mere £10.00.

***Touchline:** Incentive Software Ltd, Zephyr One, Calleva Park, Aldermaston, Berks, RG7 4QW. **Tel:** 07356 77288.*

Stop Press!

Database Software have just acquired the rights to two well established DTP packages for the C64 and C128. Both titles originate from AMS. The titles are Stop Press, and Stop Press with mouse, costing £39.99 and £69.99 respectively. Productivity software is a little thin on the ground for the 64/128, so well done Database.

***Touchline:** Database Software, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP. **Tel:** 0625 878888.*

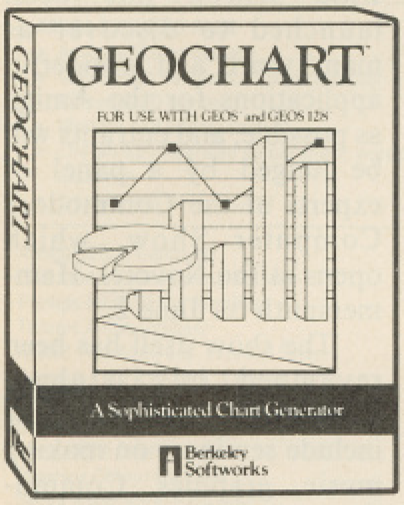
Viva Geos

There is no quicker or easier way to present data than with a graph or chart, and GeoChart is a simple and inexpensive way to produce good-looking graphs and charts.....etc etc', or so the blurb goes. Berkley Softworks have added yet another program to their ever growing army of Geos utilities. GeoChart is ideal for anyone that wants professional-looking documents and charts, but can't spare the time needed to plan and draw graphic illustrations.

Like all the Geos appli-

cations, GeoChart is the 'What you see, is what you get' format. That is, pull down menus, windows and icons. GeoChart is for the C64 and C128 owners.

***Touchline:** Berkley Softworks, 2150 Shattuck Avenue, Berkley, California 94704. **Tel:** 415 644 0883.*

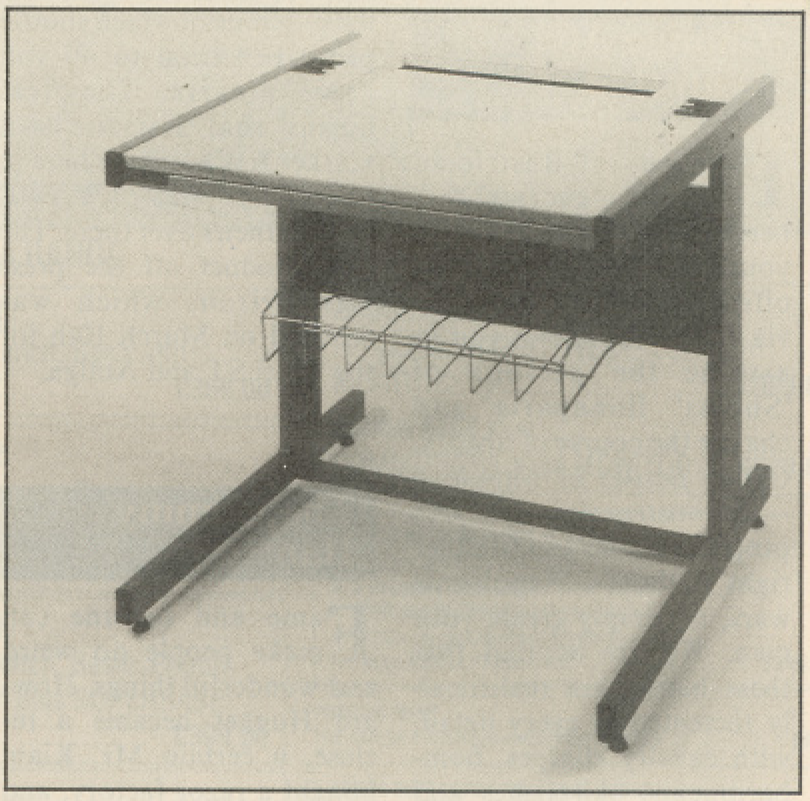


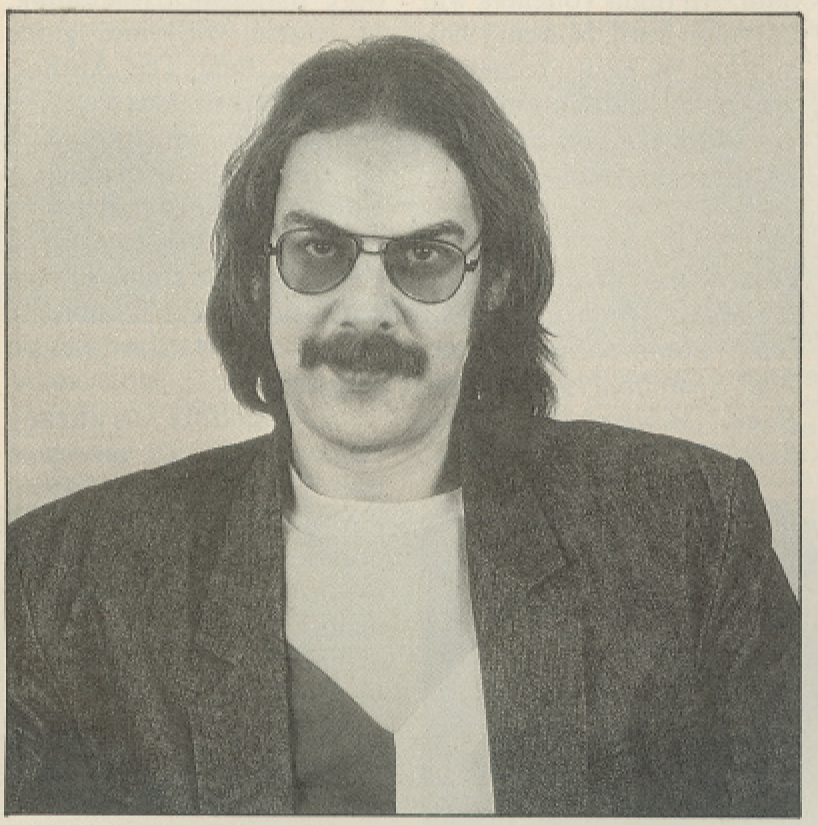
Hubba Hubba!

Electronic Arts have enhanced that popular game from Bullfrog, *Populous*, by adding a new sound track from Rob Hubbard.

Rob, who left the UK in '88 to work in sunny California, has since been working on a major project. The work involves creating a code package that will allow real soundtracks to be produced. The driver, we are told, will be usable over and over again for any product that requires a soundtrack.

***Touchline:** Electronic Arts, Langley Business Centre, 11/49 Station Road, Langley, Berkshire, SL3 8YN. **Tel:** 0753 49442.*





Rob Hubbard – artist in exile

Make a Stand

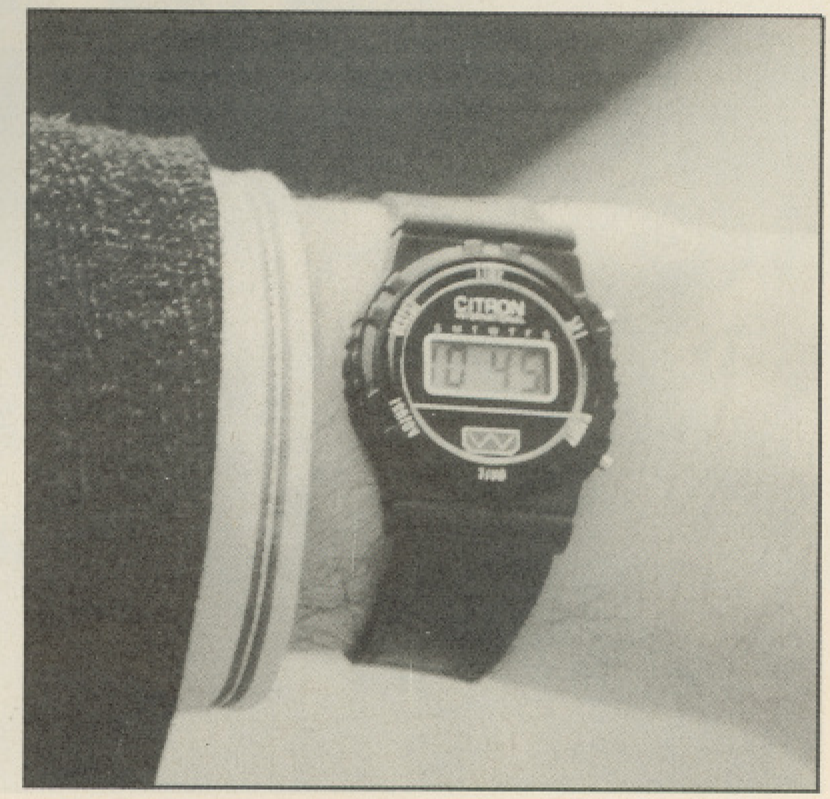
Have you noticed how computer furniture never gets a mention? Well, here goes. From MDS of Hemel Hempstead comes the MDPS 750 printer stand. Of all the peripherals, printers seem to cause the most positional headaches, but this new stand could solve all your problems (although the £204 price tag may put you off). It is, and I quote "uniquely designed to achieve maximum efficiency, and part of a comprehensive modular desk system which measures 750mm wide, 710mm high and 750mm deep, hs an 18 inch cut-out in the top, making the unit suitable for both top and bottom feed printers". The other good point is the electronic cable management system incorporated into this unit – no more wires and cables need get in the way.

***Touchline:** MDS Industries (UK) Ltd, Factory 1, Ebberns road, Hemel Hempstead, HP3 9QS. **Tel:** 0442 231305.*

Watch Out

Microprose are about to launch an attack on your senses with *Navy Seal*, a 64 game packed with action. You are a member of the Elite US Navy force, the sea-air-land commandos, who are experts in sabotage, demolition, reconnaissance and infiltration. So that you don't miss the boat, a *Navy Seal* Digital Watch (Right) will be given away with every copy purchased. The release date should be in early April, at an expected cost of £12.95 on disk.

***Touchline:** Microprose, 2 Market place, Tetbury, Glos, GL8 8DA. **Tel:** 0665 54326.*



Change At Waterloo

Persons of Restricted Growth everywhere, here's your chance to get your own back! The latest offering from SSI, available via US Gold, allows you to assume the persona of 'Shorts' Bonaparte and change the course of history. Called *Battles of Napoleon – a construction set,* this program allows you to either right authentic Napoleonic wars, or simply create your own. We are assured that these battles are realistically recreated in every detail, with cavalry charges, bombardments, skirmishes and formations. Priced at £24.99 for the C64, the game offers good value.

***Touchline:** U.S. Gold, Units 2/3, Holford Way, Holford, Birmingham, B6 7AX.* ***Tel:** 021 356 3388.*

Special Relationship

News has reached us of a bilateral deal between Krisalis and Action Ware, the outcome of which should prove beneficial to all you game players. The deal means that Krisalis will market AW software here in the U.K., and AW will market theirs over there. The first product off the press was *Prison*, which was released on March 10th for the Atari ST and Amiga.

The Hammy Hamsters

Fame and fortune can make people do weird and wonderful things. Howard Hughes became a recluse, a certain Mr. Kiam bought a razor factory, and Mr. Wyman managed every man's dream. Now it's the turn of the Darling family. Not content with producing both passable budget software, they have recently broken out into song. To help promote their latest release, *Rock Star*, they've formed a group which, for reasons best known to themselves, is called The Hamsters. Judging by the picture. I know which Hamster I'd like as a pet!

***Touchline:** Code Masters Software Co Ltd, Lower Farm House, Stoneythorpe, Southam, Waricks, CV33 0DL.* ***Tel:** 0926 814132.*



Amiga Airlines

Buyers of Commodore's new 'Amiga Air Miles Pack' will be receiving free air miles along with their machines. The package offers an Amiga 500, a TV modulator, three new games (*Roger Rabbit, Nebulus* and *StarRay*), comprehensive paint packages and 500 free air miles, all for £499.99. The offer will give Commodore customers the chance to book return flights to a number of European cities, including Paris and Amsterdam. All flight vouchers will be valid until March 1990, and can be easily redeemed via the special Commodore Air Miles Hotline, as well as local travel agents.

***Touchline:** Commodore Business Machines (UK) Ltd., Commodore House, The Switchback, Gardner Road, Maidenhead, Berks SL6 7XA.* ***Tel:** (0628) 71456.*

Hit The Jackpot At The Commodore Show!

Commodore are giving away £500 worth of computer equipment to the Amiga user with the most innovative and unusual use for his machine. A nationwide search has been launched to discover as many weird and wonderful applications for the Amiga as possible, and entrants will be judged by a panel of experts at the Commodore Computer Show, which opens at the Novotel, Hammersmith on June 2.

The show itself has been revamped by Database Exhibitions, and will now include seminars on making music, graphics, Commodore magazines (including, of course, *YC*) and game writing. 'Beat The Author' competitions are also planned, in which punters will be able to take on 'named' authors and try to beat them at their own game. But one of the big highlights of the show will undoubtedly be the innovation competition.

Commodore has already discovered a wide variety of uses for the Amiga, from bird-watching to betting on horses, and has even heard of one user who works out menus for reptiles on his machine! If you think you can match these innovative applications, all you need to do is write a brief description – no more than 500 words – and send it in, along with any necessary support material on disk, to:

The Editor,
Your Commodore,
Argus House,
Boundary Way,
Hemel Hempstead HP2 7ST.

Entries should reach us before May 31st.

Mailbag

Your chance to air your views in Your Commodore

I have just received a Commodore PET 3032 Computer from a friend. With it was a 8050 Disk Drive and 4022(P) Printer. I was wondering if you or any of your readers had any information on software availability. Please send any information to me, thank you.
R.M. Crock, 29 Grange Road, Bromley Cross, Bolton, BL7 9AU

I was recently given a PET 8032, a 8050 Dual Drive and a 4023 Printer together with four good software packages complete with manuals and dongles, where necessary. I only want to use the PET for wordprocessing, and so I am sending out an appeal. Is there anyone out there that has a program for the 8032 that they don't want? If so, please contact me at my address. Thank you.
J. Simpson, 4 Goldenstones Avenue, Dunbar, East Lothian

Come on now everyone, we know these are pretty ancient machines, but there must be someone out there who can help!

Dear YC

Is there anything that can be done with old printer ribbons? I feel that it's a great waste to throw them away.

Could one possibly revitalise them with ink in some way? If so, which ink should be used, or can I send them somewhere to be done?
J. Osborn, Luton, Beds

Our reply
The answer, John, is yes. Printer ribbons can be re-inked. If you get a copy of Micro Computer Mart, *you will find a firm advertising this service every week.*

Dear YC

With three CDU disks, I have had some loading problems. The fault lies not with yourselves, but rather with the newsagents, when they stack magazines on top of each other. I have found that 'Validating' the disk beforehand seems to cure this problem.

I don't know what proportion of failures are returned, but this Tip may help in some way.
J.F. Paddock Wood, Tonbridge, Kent

Our reply
Thanks for this helpful little tip. We do get some returned disks, and many of them are cured by this simple method.

Unfortunately, newsagents and booksellers do not realise how delicate disks are.

Dear YC

I have been computing ever since the Commodore 64 was first produced. In this time I have bought various computer magazines, mostly multi-format, and I've noticed various discrepancies between reviews of the same game. For instance, Tetris received 50 out of 100 from one, but 97.5 from another.

The trouble with two completely different results is which one do I believe? Do I take the advice of magazine A and stay clear of it, or do I take B's advice and sell my TV so that I can buy it?

The best advice for anyone is to try and get into the reviewer's mind. Some reviewers set their hopes too high, and so if a game doesn't come up to their expectations, they give it a poor write up.

A recent review of *Thunderblade* is a good example of this. One reviewer of this game said – "The graphics at this point, Level 1 stage 1, were very disappointing – the buildings are just stacked squares".

Personally, I think the graphics are very good. Okay, the first stage on each level is blocky, but you can't really expect hi-res graphics can you? It would be much too slow. Anyway, you've always got stages 2 and 3 in each level to make up for it.

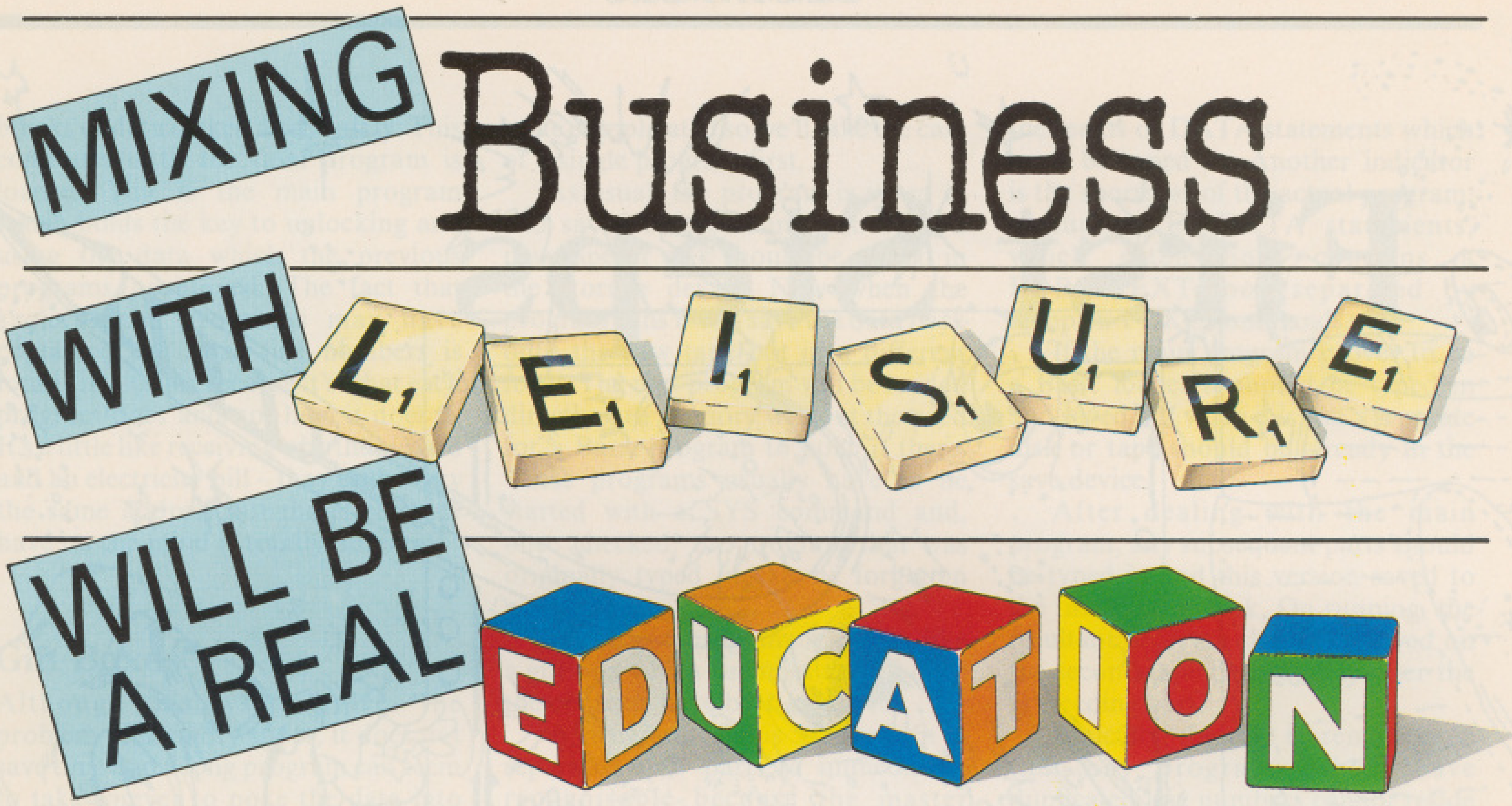
Another annoying item is 'Moaning'. You know the kind of thing – "The Q button doesn't work, so you have to use the F7 key". A great example of this can be found in the same review of *Thunderblade*. The reviewer said 'Unfortunately, it is hard to find any difference between cannon and air-to-ground rockets.' There are two possible reasons for this – (A) His/Her cassette deck/disk drive was faulty or (B) their copy of the game was faulty. These are the only possibilities, as my copy of the game is the opposite.

In conclusion then, what do you do if two reviews differ? The first option is to see if you can have a demonstration of the program, therefore allowing you to make up your own mind. The second option is to simply take a chance.
N. Kingsley, Chichester, West Sussex

Our reply
Thanks for such a long and 'critical' letter. However, I think you're missing the point (as do a lot of people). Reviewers of software do not differ from any other form of reviewer, be it Films, Videos, Records, Food, Clothes Etc Etc.

The whole point here is this – the comments passed are the comments of that particular person, NOT the comments of a 'professional critic'. For example, I personally have never liked Elite, *but for 1.5 million other users it is the best program ever.*

Likewise, for me Infocom are the only people that produce decent adventures, but I bet thousands of people would not agree. Basically, one must remember that a reviewer's comments are his own personal comments. A review only really tells you what the objective is and how the game plays. There is only one person that can say whether it is good or bad, and that is you.



Commodore computer show

Britain's brightest event for Commodore computer users is back! And there's more to see than ever before.

This show has three main themes covering some of the major uses to which Commodore machines are put. There are over 70 key companies who will be exhibiting their latest products, which means that just about everything that's new in the Commodore world will be on show!

Business

Many companies will be demonstrating their latest software and hardware, specially designed to release the full business potential of Commodore computers.

As well as products for the C64 and Amiga series, you'll be able to try out applications for the price-beating Commodore PC compatible micros.

And you'll also be able to attend seminars covering all aspects of using Commodore micros in your business.

Leisure

The C64 and Amiga computers are the most powerful 8 and 16 bit micros for producing fast-action arcade quality games. The range of new software on show will demonstrate how these machines' power is continually being stretched, producing faster and even more addictive games with superb graphics.

If you're a keen game player, you'll find there's so much on offer at the show you're guaranteed a real treat!

Education

Commodore micros are now used as educational tools all over the country. With the development of BBC Basic on the Amiga, and the advent of Desktop Video (combining TV pictures with text and graphics), the range of educational applications is endless.

At the show you'll see how the latest software packages are making real breakthroughs in the educational sector, and be able to try them out for yourself.

Novotel Exhibition Complex, Hammersmith, London W6

Friday to Sunday
June 2 to 4

10am-6pm Friday & Saturday; 10am-4pm Sunday

Special Events

As well as special events and presentations, you'll also be able to meet some of your favourite celebrities, and maybe get a chance to talk with them about how they use micros in their work.

So for a great day out, whether you want to see what the future holds for Commodore computers, to buy the latest software or to get advice on specific applications, the Commodore show is the place to go. And if you send in the coupon today, we'll knock £1 off the price of each ticket!

● ***For the first time we are offering a family ticket for just £11 allowing entry for two adults and two children – saving up to £7 off the usual entry price!***

How To Get There
By Underground: *Hammersmith (Piccadilly, Metropolitan & District).*
By Bus: *266, 714, 716, 290, 30, 72, 73, 74.*
Car parking facilities available at the Novotel.

Advanced ticket order Commodore computer show

POST TO: ***Commodore Show Tickets, PO Box 2, Ellesmere Port, South Wirral, L65 3EA.***

Please supply:

☐ Adult tickets at £4 (save £1) .... £ ______
☐ Under 16s tickets at £2.50 (save £1) ..... £ ______
☐ Family ticket at £11 (save £7)................ £ ______
Total £ ______

☐ Cheque payable to Database Exhibitions
☐ Please debit my Access/Visa card no:
|_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|
Expiry date: [/]

Signed..

Admission at door: £5 (adults), £3.50 (under 16s)

Advance ticket orders must be received by Wednesday, May 24

Name..
Address..
....
....Postcode ....

PHONE ORDERS: Ring Show Hotline: 051-357 2961
PRESTEL ORDERS: KEY *89, THEN 614568383
MICROLINK/TELECOM GOLD ORDERS: 72:MAG001
Please quote credit card number and full address

A693

First Steps



How to make sense of Listings **By Norman Doyle**

Typing listings in can be much more of a trauma than it need be. At *Your Commodore*, we are aware that sometimes the process seems difficult or even nonsensical. Our Listings with their Syntax Checker and special codings may ease the way, but sometimes more support is required.

Many programs are listed in several parts and each part uses the same line numbers. How is this possible and how do you set about typing them in?

First of all it should be explained that a set of these programs form a suite which operate in unison. The way to use them is to type in the first program and save it. Now the computer should be cleared by entering NEW or even by switching the computer off and on.

Proceeding to the second listing, the program is again typed in and saved. If you're storing the programs on tape, you should ensure that this program is saved immediately after the first one. For disk users, it doesn't matter where it's saved as long as it's on the same disk as the first program. After checking that all has been saved safely, the machine is again refreshed using NEW or the power switch.

Any subsequent programs should be treated in the same way as the second listing and the result will be a string of programs. One word of caution which should be heeded is that the programs should be saved under the name given in *Your Commodore* because each program occasionally uses one of these names to load in the next program in the string.

Suite Confusion

One question which we're often asked is why these multipart programs are needed. The answer is that they are a memory-saving or time-saving measure.

Many programs use redefined characters, machine code patches and sprites. The information for this has to be poked into position because they rarely occupy memory locations which Basic uses. This difficulty can be overcome in either of two ways.

The data can be poked into position each time the program is loaded and forms an integral part of the program. This has the dual disadvantage of the time the program takes to move the data in but, more importantly, reduces the amount of space for the master program itself.

Another way of achieving the same result is to use a series of set-up programs before the main program is loaded. The set-ups are loaded and run in turn and each one pokes information into memory where it cannot be touched by any subsequent loading actions. Consequently, once the program has done its job it is no longer needed and the program can even be removed by typing in NEW without ruining the effect of the main program when it is eventually loaded.

One thing that you cannot do is to switch the computer off and on again! This action totally clears the memory, including the data which has been poked in.

When the first program has done its job, the next program is loaded normally and run. Now there are two

blocks of data locked away safely. This continues until the final program is loaded. This is the main program which holds the key to unlocking and using the data which the previous programs positioned. The fact that each of the programs may have contained the same line numbers is totally irrelevant because it's what each program does and not how it does it. It's a little like receiving a birthday card and an electricity bill – they both carry the same address but the effect they have on the mind is totally different!

Gift Boxes

Although this system solves the problem of memory space, it does not save time and a long program can seem to take forever to poke the data into position. One way to overcome this is to create a program which not only pokes the data into position, but also saves the pure data block afterwards.

These *throwaway* programs, or Basic loaders to give them their correct name, are quite common and need treating separately. The process can be quite complicated so we'll take the case of a single program first.

As usual, the program is typed in and saved before running it. Next a new tape or disk should be placed in the storage device. Now when the program runs it will save the data back onto the new tape but in a different form. The new program will be loaded directly into memory without the need for a Basic program to poke it there. These programs usually have to be started with a SYS command and, once checked, the program that was originally typed in can be forgotten and erased.

The Basic program is rather like a gift box which becomes useless once the contents have been removed.

A program of this kind which is separated into parts is immediately recognisable because the master program is the first in the chain. If this is in Basic it should be typed in and saved. If it is a Basic loader it must be treated differently.

Spot The Loader

A Basic loader can be recognised by the reams of DATA statements which must be typed in. Another indicator is the shortness of the actual program, excluding the DATA statements, which often only contains a FOR...NEXT loop separated by complex POKE commands.

If the main program proves to be a Basic loader it should be typed in and saved but when it is run a separate disk or tape should be already in the save device.

After dealing with the main program, any subsequent parts should be typed in and this version saved to the first tape or disk. On running, the resultant program should be saved on the second tape immediately after the preceeding part.

Basically, the rule to remember is that any programs which have duplicated line numbers cannot reside in memory at the same time. They should be typed in and saved separately according to the instructions given in the relevant *Your Commodore* article or the REM statements in the listings themselves. Stick to this rule and you'll never go wrong...well, rarely!

The Case of the Electronic Home

Leaving aside his relentless search for the dreaded Moriarty, Mr Sherlock Holmes and his old friend Dr Watson visit the Ideal Homes exhibition and discover that 221B Baker Street is not all that it should be...

By Gordon Hamlett

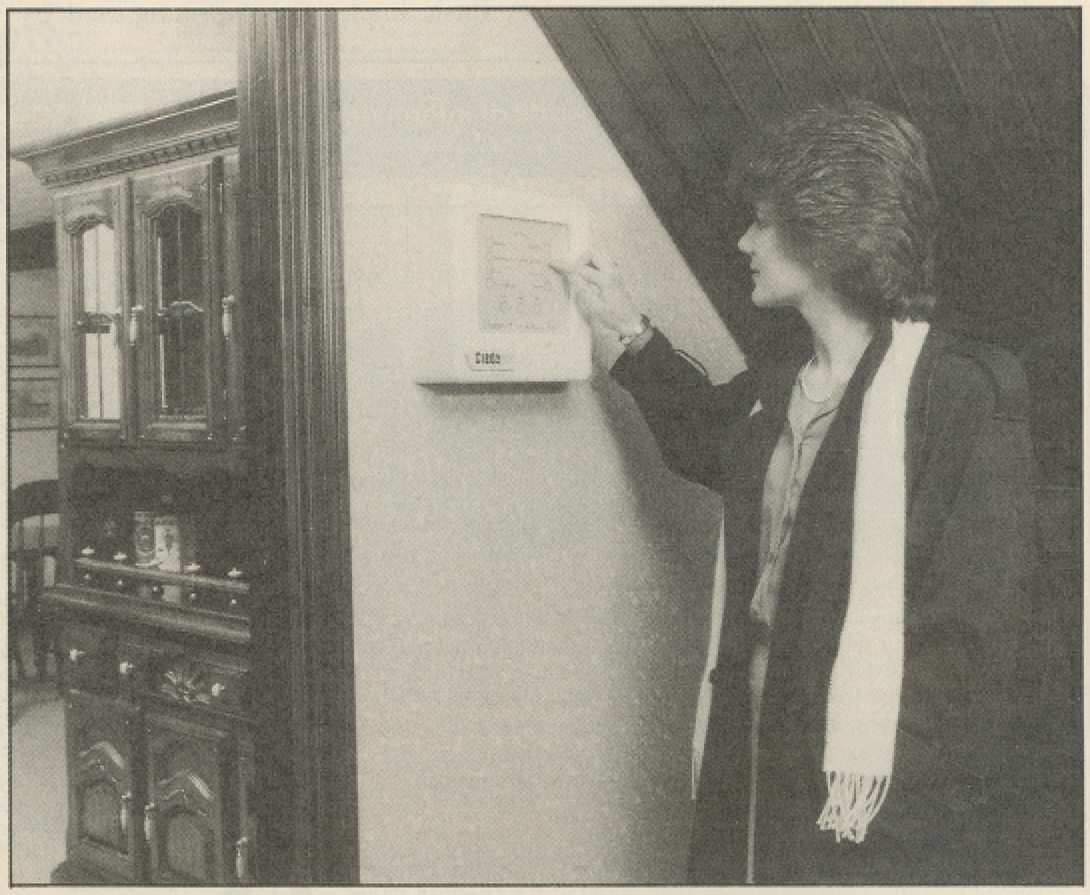
'Holmes, I've got a letter here from a distraught magazine editor asking us to investigate the future of the electronic house, but I don't know how to go about the job.' 'Well Watson, you can always pop down to the furniture exhibition at Earl's Court and see what's new.' 'That's Ideal Holmes!' (*Oh God – Ed*).

So it was that our two intrepid detectives went off in search of clues as to how long it would be before everything in the house could be operated at the flick of a switch.

When they arrived at the show though, they were in for something of a surprise. Whereas even a couple of years before, every machine was being heavily promoted as the all singing, all dancing model with more lights and buttons than you knew what to do with, this year's displays were much more minimalist in design. It was as if it was assumed that the machine could do everything that was asked of it, and it was simply left at that.

'There are two distinct problems here' said Holmes. 'To start with, you will notice that just about every machine, from cameras to washing machines, has chips in it, which in turn gives the end user access to a plethora of functions. But although the customer may appreciate having all these extra facilities, he doesn't actually use many of them. After all, just about every single compact disc player lets you program the tracks in any order that you want. But who in their right mind is going to mess about with the order of the movements in a Mozart symphony?'

'The second problem is that nothing is compatible unless you stick to just one manufacturer. So the customer ends up with separate remote control handsets for his television, video and hi-fi. None of them will turn on his dishwasher, and he can only use them over a limited range. What is needed is some all-powerful control system.



'I notice that all the display houses have a study' commented Watson. 'The idea of people working more and more from home persists, even though there is little evidence to suggest that it is actually happening. It seems that computers are still regarded as toys unless they possess those three magic letters – IBM. Modems still have not caught on as they have done in America. Instead, no home should be without its own personal fax machine.'

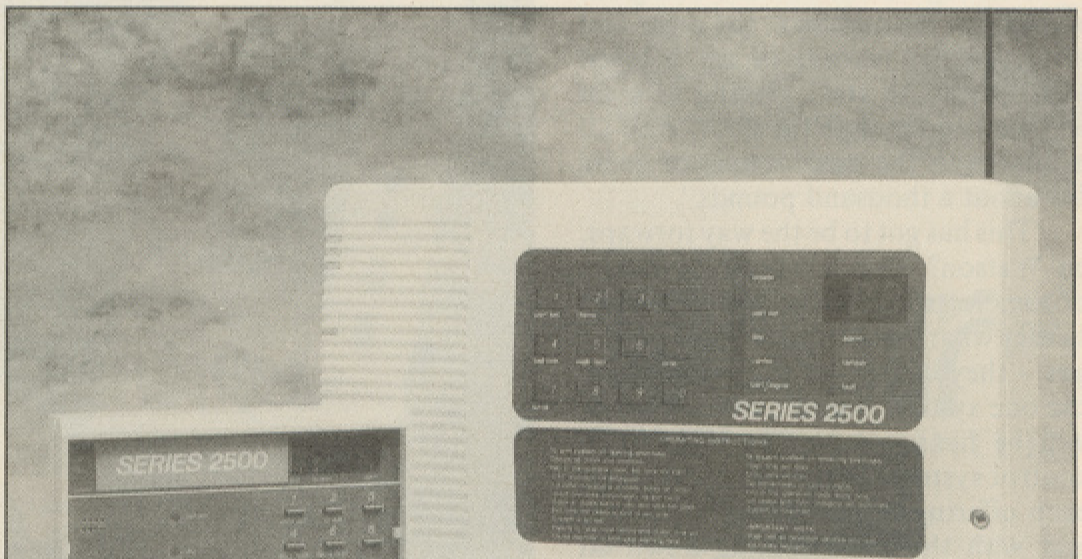
'If we are not very careful Watson, you and I could soon be out of a job. The one major area of expansion in the electronic field is in security systems. I see that Modern Alarms are offering a wire-free programmable system – easy to install, no messing up the decoration, and it can be taken



with you when you move. And every type of alarm imaginable is here – magnetic contacts, heat, radar, sound, vibration, breaking glass, infra red beams and close circuit wiring. There are even personal alarm buttons.'

'How is the enterprising burglar supposed to get past that little lot, and if there are no criminals, then what are we to do? Ah, here is something that would fool even me. A machine that emits artificial barking noises. Remember that case of the dog that didn't bark in the night?' This device would have totally messed up that story.'

'I can find no trace of robotics here Holmes. It looks as if we will have to put up with Mrs Hudson's housekeeping for at least another year. This is one field that has definitely not progressed as quickly as everybody anticipated, although I still hear rumours that two of the biggest fast food chains are battling to be the first



to get their customers served by automatons.'

'At last, Watson, I think that we have found what we were looking for. A system that will control every domestic appliance in the house and which can be operated either from the central processing unit, from a handset or, more importantly, via a telephone link. Just think of it. Being able to phone home if we are out on a case and instruct the video to record *Crimewatch* and *Police 5*.'

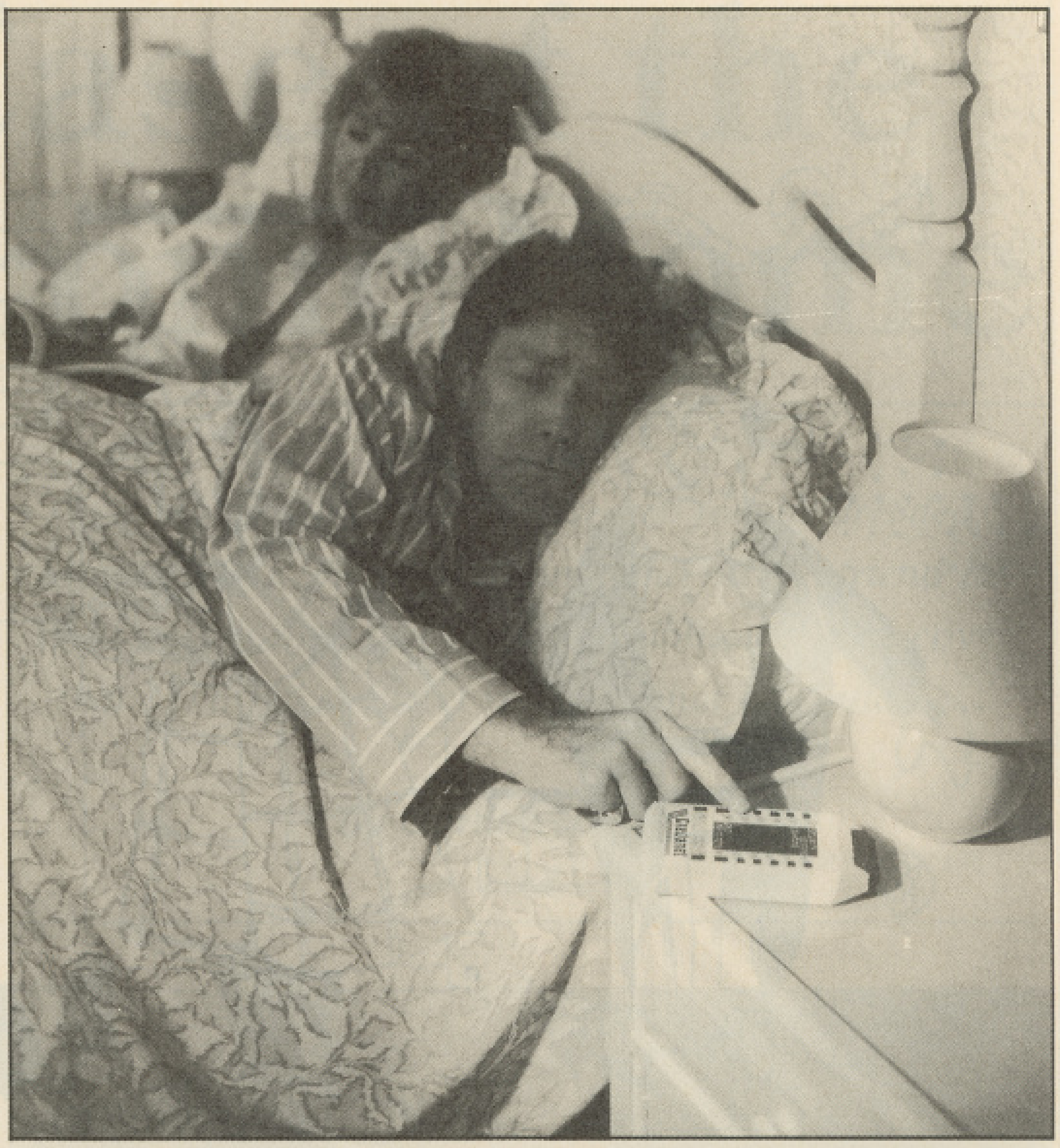
'I see that the system – Credanet from Creda – works by means of a series of transceivers placed in every appliance. It then communicates with the central processing unit using existing mains circuitry. This c.p.u. has a large display panel as well, so that you keep addresses and diary dates as well as being able to check up on share prices and train times. That would have been useful in the Baskerville case.'

'We would be able to control the temperature in every room in the house, varying it as and when we saw fit. Lighting could be similarly controlled, ranging from individual lamps upwards. Imagine being able to turn out the kitchen light from your bed when you had forgotten to do so. All your aforementioned security devices could be checked easily, although it would lessen the impact of my detective powers if I knew beforehand exactly who had turned up at my front door.'

'Turning on the cooker, doing the washing in the middle of the night and arranging for records of my beloved violin music to start playing as soon as I walked through the door. And all available in the next couple of years for about a thousand pounds.'

'This has got to be the way forward, eh Watson? This is where the future of the electronic house lies. It doesn't matter what make of appliances you've got – they can all be controlled from the one system. I would guess that we will be hearing of many more such control systems in the next few years, each offering more and more features and with the price coming down all the time.'

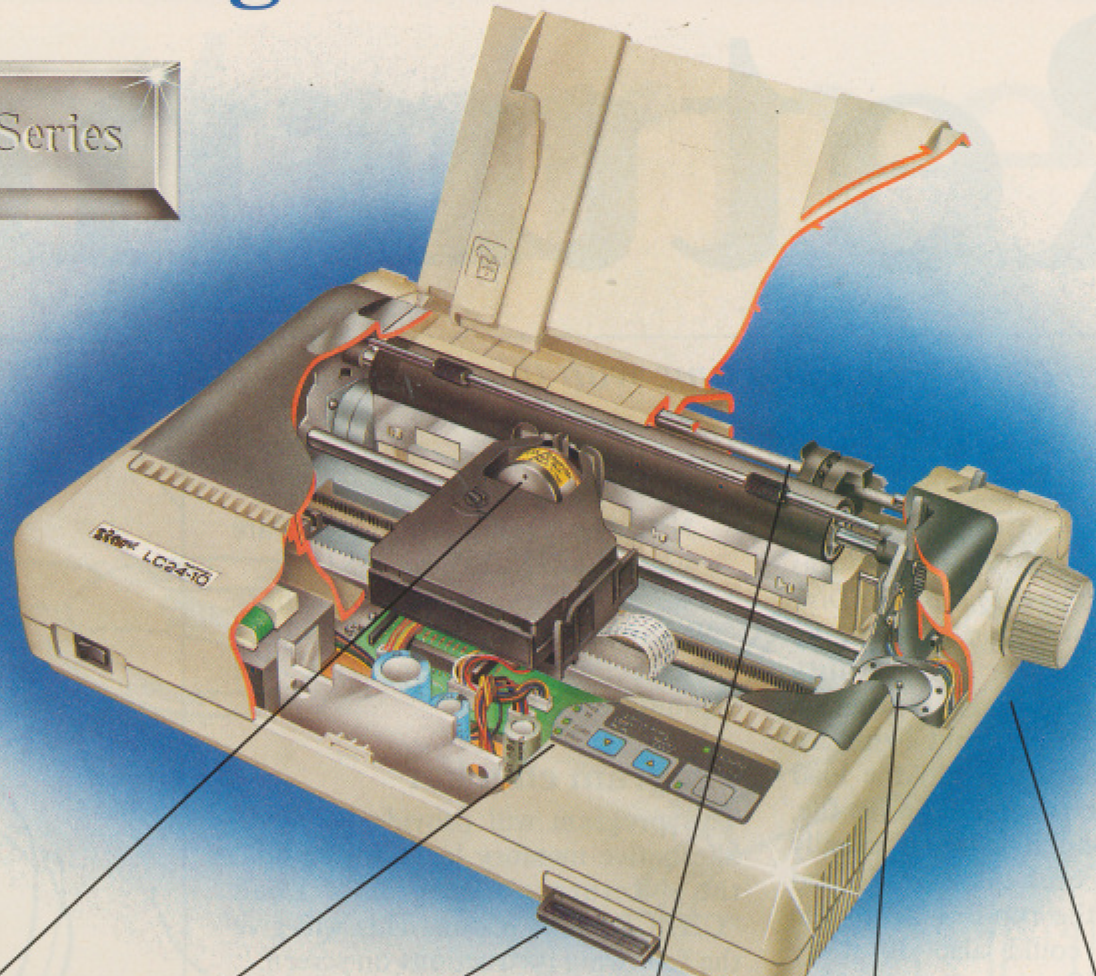
'There's only one thing left for us to investigate now Watson, and that is to find out what is on the other side of that yellow door.' What's so special about the yellow door Holmes?' 'It's a lemon entry my dear Watson!' (*Have mercy, Hamlett, no more please* – a distraught Ed).





Quality. Designed from the inside out.

The Business Series



£ Value — 24 24 Pin — Simplicity — 8 A-Z Fonts — P Paper Parking — 170 Speed — Compatibility — Support

For state-of-the-art printing the multi-font LC24-10 gives you value for money as well as the superior quality of a 24-pin printhead.

The LC24-10 is compatible with almost every computer's needs whatever the system. It can emulate most industry standards with ease, a standard feature that we think should always be included. For connection over longer distances, there is even the option of a low-cost serial converter.

Putting the LC24-10 to work couldn't be simpler. At the mere touch of a button on the front control panel you can choose any one of eight resident letter quality fonts, determine your



pitch size or simply put the paper park into action. (No more inconvenient DIP switches to hamper you). As with all our printers, it offers both double and quad height print for striking headers.

170 From 0-170 characters in one second and that's just the elite draft. The LC24-10 will also perform word perfectly at a rapid 57 cps producing exceptional letter quality for pristine correspondence and important documents. And while you print, the 7K text buffer frees you and your PC to get on with other things.

£ No other 24-pin printer in this price range can match it. Features like its generous text buffer, expandable to 39K, built-in push tractor and revolutionary continuous paper parking facility are all included as standard. Another impressive feature you won't find as an optional extra is the LC24-10's versatile range of print variations. All fonts come with the option of shadow, outline and even shadowed outline to turn all text into works of art.

Behind every Star printer there's a guarantee of superb reliability together with a comprehensive back-up service for peace of mind. Should you need any help at any time, you can rely on our national three-level support system which operates throughout the dealer network and is backed by the finest distributors in the country.



THE star COMPUTER PRINTERS

Star Micronics U.K. Ltd., Craven House, 40 Uxbridge Road, Ealing, London W5 2BS. Telephone: 01-840 1800.

A Division of Star Micronics Co., Ltd., Japan.

Please send me all the information I'll need. (Alternatively, just call Belinda on 01-840 1829).

Name ________ *Company* ________

Address ________

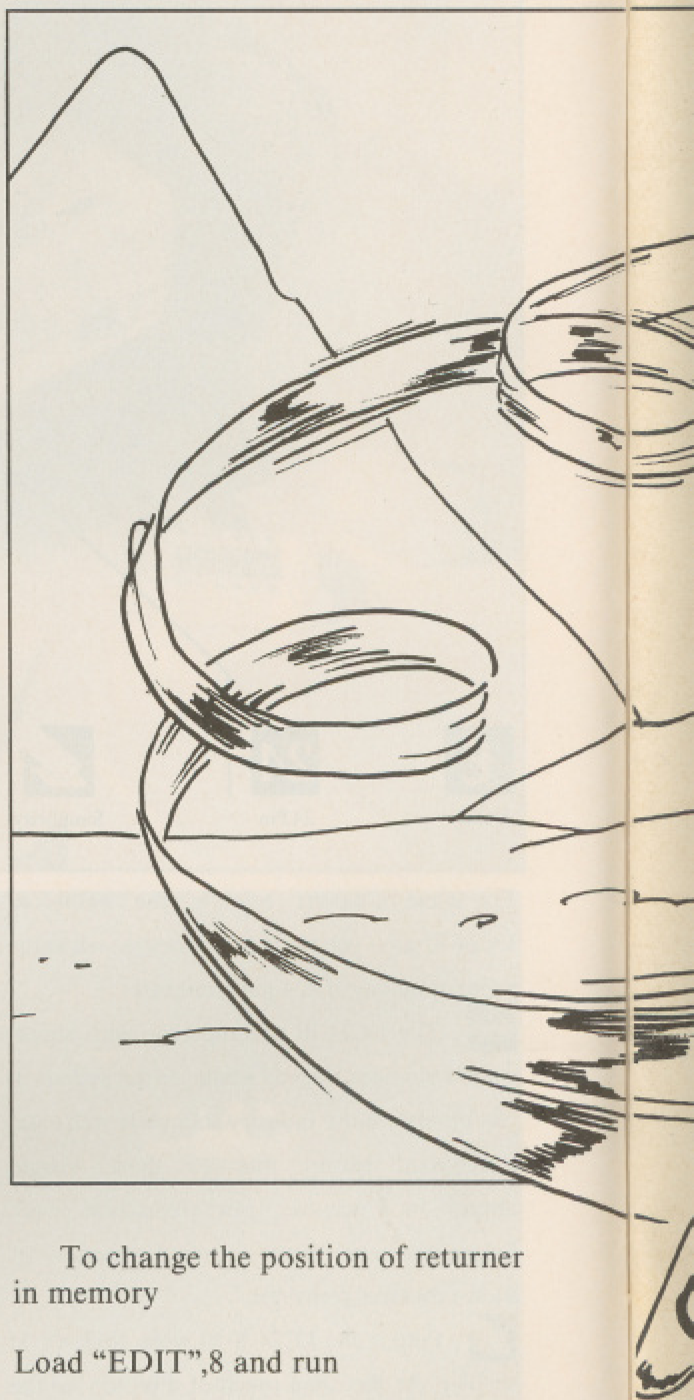
________ *Postcode* ________ *Telephone* ________

L24YC5

All trade marks acknowledged. HMSO *Public sector customers may purchase at preferential terms from HMSO. Contact 0603 695256.*

Returner

Return to the menu at any time with this ingenious program



The problem of keeping records of what's on your disk was addressed in *Commodore Disk User* Vol 1, No 2 (Jan/Feb 88) by Menu Maker. This useful utility allowed you to load any program direct from the Menu, but to return to MENU you had to add the load statement to each program on the file, which is easy enough to do, but usually means putting check questions in and having to end the running up of the program. You could also do it by pressing Run Stop and reloading "Menu", but that's rather a waste of time.

The program Returner allows you to leave a program at any point or time, and return to the menu to select an alternative program without all the problems outlined above.

On loading and activating the 'RETURNER' program code, the first job that it does is to set up the 'RESTORE' key, and automatically load a MENU.

Obviously, for the program to work correctly, you must have a 'MENU' program on your disk. We have published quite a few such programs in the past. Alternatively, you could write one of your own. To get the best results, it is advisable to have at least two, if not more, programs on your menu.

Once you have your menu program running, yu select which program you want as normal. On running the selected program, you can press the 'RESTORE' key at any time to return to your MENU program.

If the returner won't work

There are two reasons why Returner may not work:

1. The program in memory is using the NMI interrupt, or it is restoring the interrupt vectors to normal values. You need to alter your program (if possible) so that it's not using the NMI interrupt.

N.B. The NMI interrupt values are located at 792-793 ($318-$319).

2. The memory location where 'Returner' is located is being used by the program being run. See solution below.

Program checker

This program will provide you with alternative memory locations to hold the 'Returner' program. Load "CHECKER",8 & run. This will give the following instructions on screen.

First Type in N
Then Load Menu, Load A File, & Reset
Load Menu, Load Another File, Reset
Repeat Until all Files Loaded
Load Checker & Type Y
Have You Wiped Before?

When running, 'Checker' will show a blank screen until processing is complete. This will take approximately three minutes and then show on the screen:

Ready
Load "Menu",8

Press Return. Run the menu, then load first file, then reset computer by pressing RUNSTOP & RESTORE (or by using a Reset cartridge) **Do not switch the computer off & on.**

Reload the menu, run the next program, then reset as before and continue until all files have been loaded. Once all the programs on the disk have been run, reload CHECKER, run, and type in Y to the question.

Option is given to output to printer as well as screen. The result of the program is a list of free memory positions available for insertion of the program Returner. You can select the desired position to place Returner (if no space available it will tell you so, i.e. Returner cannot be accommodated on that disk.)

To change the position of returner in memory

Load "EDIT",8 and run

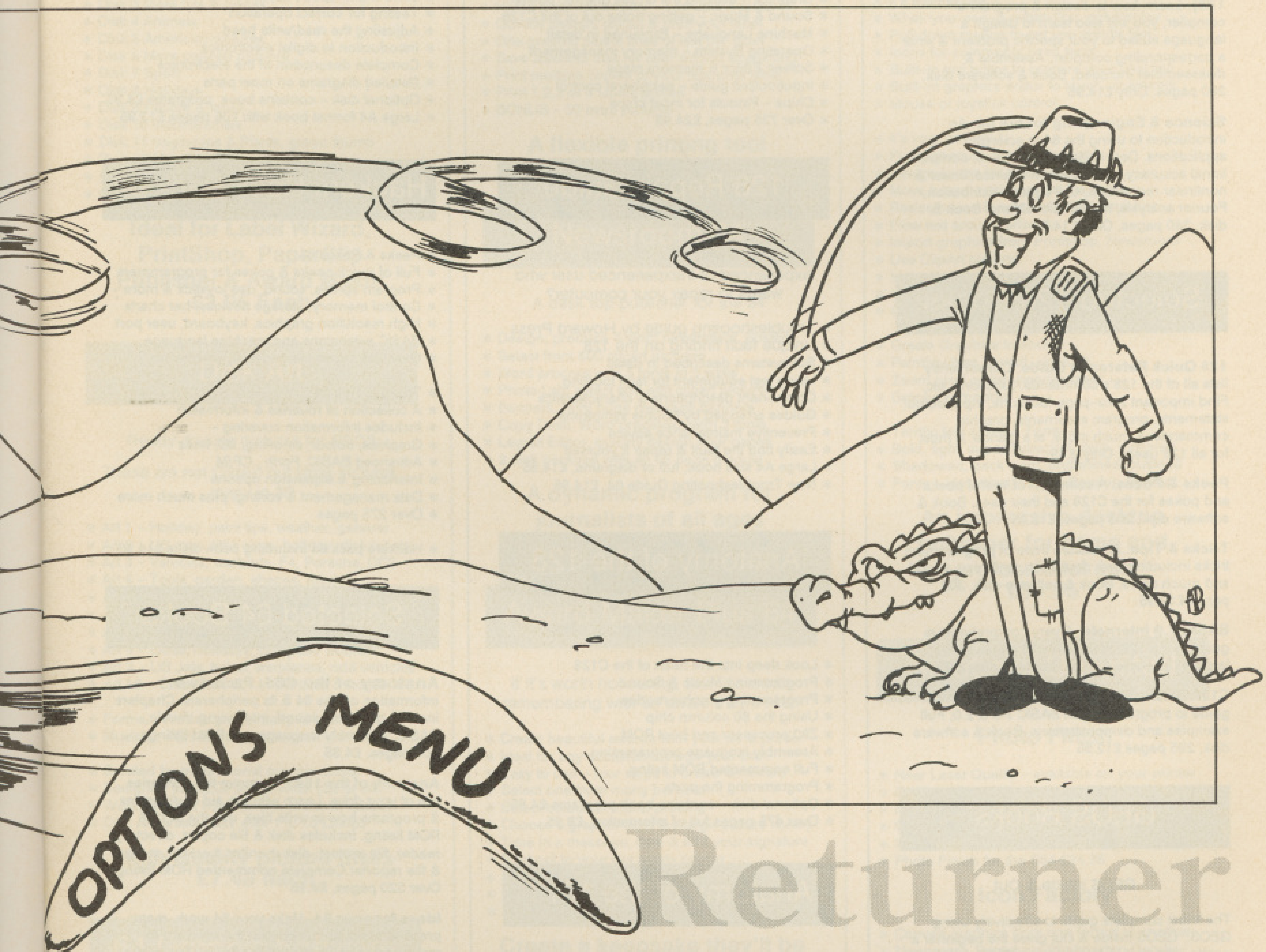
The screen will ask for the new start address. You can enter it in Hex or Decimal numbers (the Checker program provides location values in decimal).

Once entered, press return.

The screen will then ask you to state which file you want to be automatically loaded when you press the RESTORE key. If not MENU, then alter to your own requirements – press Return.

The screen will then ask what name you wish to give the Returner program. Put your own name in. The screen will then ask "Are You Sure?"

Returner



Returner

– if not, type N, if you are, type Y. N returns you to the beginning of the screen, and you will have to re-input memory location etc. When you type Y, it will save the newly-named program to disk.

Starting up your computer

When you start up your computer, you initially load and run the program you named and saved above. It will automatially install the RESTORE key function and run the program you asked to be first loaded (see above). I recommend that it is the menu as pressing the RESTORE key will automatically return you to this program until the computer is switched off.

Note: A good place to put the 'Restorer' program in memory is 820 ($334), as it is not normally used when using disk. It relates to the tape drive use, so you won't be able to use the tape drive.

If you have a multipart game, Restorer can be used to restart the game by asking it to automatically load in the first part of the game as part of its program (see above). The start program must, however, have a Basic line number at the beginning e.g. 10 SYS3000.

Addition to Returner instructions

It is not recommended to press RESTORE during the use of the disk drive or printer, as the menu may not work properly if you do so.

If you need to press RESTORE while the printer or disk is in use and the menu doesn't work properly, then just press RESTORE again and the menu should work properly.

Oxford Basic

Despite its name, this is a compiler with several very interesting extras

By Norman Doyle

The Oxford Basic package is a suite of four programs designed to get the best from Basic and beyond. The main part of the program is a stunning compiler but, before using this section, the program has to be optimised.

At the beginning of the whole process, a good set of utilities can speed things up and keep it all neat. Oxford Basic's Toolkit fits into the high memory slot at 49152, and offers ten new commands which are outlined elsewhere.

One glaring omission is an automatic line-numbering command, and another is a command to undo the execution of a NEW or a recover a program after a reset. It's only my opinion of course, but to me these missing commands would have enhanced an otherwise comprehensive package.

Vital Statistics

The Analyzer really gets inside a program and squeezes out every possible value and measurement that the programmers could think about. It takes a little time for the program to be read in, but the wait is worthwhile, because there are seven options available for analysis afterwards.

A Flow Cross-Reference revealed the following facts about the program:

```
STATS REPORT FOR DISK:-
                                  SEG TOTALS
PROGRAM LENGTH ............          2201

VAR TYPE           COUNT   TOTAL
STRING              10       70
REAL                12       84

TOTAL VARIABLE SPACE .......         154

ARRAY           ELEMENTS    SIZE
BM$(35)             36       108

TOTAL ARRAY SPACE ..........         108
```

The first number in the GOSUB section is the line that is accessed and the line numbers that follow are the lines which call up the routine.

When a Data Cross-Reference is called, the variables are listed, followed by the line numbers where they appear:

```
DATA CROSS REF REPORT FOR DISK:-

ARRAYS:-

BM$(   :   460   510   710   760   820
           830   850   860   870   880
           900   910   920   930   950
           960
STRING VARIABLES:-

A$     :   110   170   190   210   220
           230   260   280   310   330
           350   370   480   510   520
           730   740   750   770   830
           880   930
B$     :   380   420
DN$    :   170   540   560   580
E$     :   130   140
F$     :   210   240   370   390   420
           600
K$     :   640
N$     :   290   310   410
P$     :   230   420
S$     :   280   420
T$     :   260   420
REAL VARIABLES:-

A      :   300   330   480   490   510
           550   570   820   830   850
           860   870   880   900   910
           920   930   950   960
B      :   580   600   620
BF     :   710   720   740
BL     :   350   370   380
BM     :   510   710
BT     :   370   460   700   740
E      :   130   140
F      :   230   240
PT     :   600   690
S      :   130   140
ST     :   200   610   650
T      :   130   140

LOOP VARIABLES:-

A
```

As a program is developed, it's easy to forget to delete lines which are no longer needed. The Deadwood Analysis cleverly scans through the program to find lines that will never be accessed. The offending lines are then listed, and a deadwood-free listing generated if required.

Another improvement can be made through the Create Header File option, which generates a short program to initialise all of the variables used in the program. The file is then saved and can be MERGEd with the main body of the program later using the Toolkit. Initialisation may seem as redundant as the LET command, but it can actually improve the speed at which the program operates.

When a variable is encountered in a program, the computer checks through the list of existing variables to see if it's a new entry, or a modification of an existing one. If the label proves to be a new one, it is tagged onto the end of the current variable list. This involves a substantial degree of memory movement and consequently wastes time.

Added to this, a variable which is commonly used may be at the end of the list, which means that the routine has to check through the full list before the current value of the desired variable is discovered. The generated header program avoids these time wasters by reserving space for the variables in descending order of frequency right at the beginning of the program.

A list of program statistics can be displayed so that memory and variable usage can be recorded. The Generate Stats option also summarises other facts such as the number of lines used, GOTOs and GOSUBs as well as the total number of variables and arrays.

```
FLOW CROSS REF REPORT FOR DISK:-
SUBROUTINES:-
 SUB      CALLING LINE(S)

 110  :   160   250   270   340   360
          500
 130  :   450   470   660
 160  :   550   570
 190  :   190   220   600

GOTO TARGETS:-
 TARGET   REFERENCING LINE(S)

 260  :   210
 430  :   200
 440  :   100
 580  :   650
 600  :   620
 630  :   610
 640  :   640
```

There is also a menu option for checking the disk directory, and to make sure that any files have been successfully saved.

The reports given by the stats generator and the flow and data cross-reference options can be saved to disk alongside the finished program. There is no direct hard copy option, but a utility file is included which will read a stored report to the printer for future reference.

Arranged Crush

The final pre-compilation routine is the Compressor, which reduces the program to its absolute minimum size. As each line is scanned, the routine removes all REMs and spaces but the compression goes beyond this. Lines are combined to squeeze in as many commands onto a line as possible.

After compression, a printout of the program may contain lines which physically span three or more screen lines. This is possible because the Compressor works with raw program code so that a command such as PRINT only occupies one byte rather than five. A line could therefore contain as many as 38 PRINT commands, which would expand to 228 characters and colons or five screen lines! This would make editing impossibly difficult, so the program must be thoroughly debugged before this operation is performed.

Although this optimises the program's memory usage, difficulties were experienced when using the compiler so I eventually decided that compression was probably better used on programs which wouldn't be improved by compilation.

Full Speed

OX-Comp is one of the best compilers that I've ever seen. It is a full-facility, four pass compiler which can cope with all of the commands in the C64's Basic library. The program also accepts all types of variables and arrays up to two dimensions. Three-dimensional arrays can be compiled, but only after the program has been pre-processed by a special utility which is included on the disk.

A compiler converts Basic code into a special form of machine code which runs up to 40 times faster than the original program. When a Basic program runs, the operating system has to scan each line in turn, turn each command into an action, and create and modify all of the variables it encounters.

A compiler scans through the Basic program and converts all of the commands and their parameters before the program is stored on disk. The conversion also means that sufficient space can be made for all of the variables which are to be used. The finished program is therefore midway between Basic and pure code.

The initial compilation can take a long time to produce, because the full program is scanned and rescanned four times before the final program is produced. During these passes, the compiler produces and erases temporary work files which adds disk access time onto the process. A long program can take up to half an hour to prepare, but the results are worth waiting for.

Wisely, the producers insist that the master disk should be used as little as possible, and a special back-up routine is included which remasters the compiler and its associated utilities onto separate disks. Unfortunately, the back-up is not the most efficient of utilities, and each sub-program is transferred individually, regardless of whether it occupies one block or 20. The result is a constant swapping of disks, which makes the whole process very laborious and unnecessarily long.

Once the copies have been made, the compilation process is automatic, and the program to be processed is placed on the same disk as the copied compiler. This can cause problems with disk space, but this problem has been anticipated by offering suitable error message generation and options to erase programs which will no longer be required for the final compilation – even the original program can be erased once compilation has progressed beyond the first pass.

The limitations of the compiler are surprisingly few and petty. Arrays must be finitely dimensioned, which means that DIM A(23) is a legal statement but DIM A(X) would be queried. If an array slips through to the compiler, which lacks an explicit dimension, the program interrupts compilation to ask for a suitable array size before continuing through the rest of the program. This may limit the efficiency of memory usage, but few programs would suffer appreciably by this insistence on explicit dimensioning.

The only commands that the compiler rejects are LIST, CONT, SAVE and RUN when it has a line number parameter. LIST is excluded because there is nothing to list in a compiled program, CONT is an acceptable omission and most people could live without commands such as RUN40 when a CLR: RESTORE: GOTO40 could be substituted instead. SAVE is harder to live without but this could be remedied by including an SYS in the original program to a suitable machine code patch.

One worrying problem which revealed itself under testing was that the maximum string length is 254 characters, instead of Basic's 255. This must surely be an oversight on the part of the programmers because it is not documented in the manual. Such errors are slightly disconcerting because they only show up at runtime, and I'd rather not sit with my fingers crossed while a program compiles hoping that another unforeseen error occurs.

All compiled programs are executed with the RUN/STOP key disabled. In most cases this is unimportant but it may be necessary to allow such an interruption so a special form of REM statement has been devised to allow the key to be enabled and disabled.

Error Tracing

If errors occur when the compiled program is being tested, it can be difficult to relate the problem back to the original program. An error which stops the execution of the program will generate a message which is similar to Basic's equivalent message, but instead of a line number a memory location is displayed.

A utility on the disk can be given this value, and it will then automatically cross-refer this and indicate the corresponding line number in the original program to assist debugging. Of course, such an error should rarely occur because any sensible programmer would have tested and

debugged the original program before compilation, wouldn't they ?

Another utility which can aid debugging was actually devised for hybrid programs which require parameters to be accessed by a machine code patch. Variables and arrays are stored differently in the compiled program so any associated code routines have to be modified. This works in favour of routines which access a current variable value, because the memory location of variable data is fixed in a compiled program. REPORT is a utility which scans the original program and prints out a table of variable locations to ease the problem of conversion.

Dotty Protection

Why, in this day and age of colour photocopiers and cameras, do software houses insist on these ridiculous colour charts? I don't mind protection systems, but I object to this form because I am red/green colour blind and the companies always seem to insist on using red, green and two other colours on these charts. Why black, white, red and yellow can't be chosen defeats me.

In this case the problem is increased because each of the four programs is individually protected. In the end, I got so annoyed that I resorted to backing up the programs using a cartridge copier – so much for protection!

Despite this, the package maintains a very high standard throughout all four sections. Maybe if the space under the ROMs had been used, the toolkit could have been more comprehensive in the manual, but no compiler is perfect and this compiler is comparable, though slightly inferior, to Supersoft's *Blitz*. What this has to offer which *Blitz* doesn't is the other three packages which make this package stand out against the opposition.

Toolkit Commands

CHANGE
DELETE
DUMP
FIND
INFO
MERGE
QUIT
RENUMBER
SIZE
TYPE

Analyzer Options

Flow cross-reference
Data cross-reference
Examine disk directory
Deadwood analysis
Create header file
Generate stats
Return to Basic

Compressor Functions

Removes spaces and REMS
Compacts program lines

Ox-Comp Facilities

Majority of Basic 2.0 commands
Four pass operation
Error checking
Location of variables printout



Line Input

Improve your 64's input with this handy utility

There are times when the standard Basic keyword "INPUT" on the C64 is a bit of a pain. In the event that a comma (",") needs to be input as part of a string "INPUT" cannot do it, because the comma is interpreted as being a delimiter between different lots of data which are supposed to be assigned to separate variables.

If you wanted to enter such a string then it has to be enclosed by quotes so that "INPUT" will read in every character. This looks silly on the screen, and is very confusing for users of the program who are not familiar with the technique – they can't understand why some text has to be enclosed in quotes in order to enter strings including commas, whereas other forms of input such as numeric data do not. It's even more confusing when the program later requires that the user use the comma to separate input into different fields!

The other slightly aggravating thing about "INPUT" is the fact that the "?" prompt is always printed on the screen prior to "INPUT", waiting for keyboard entry. This is sometimes a nuisance, since the message which asks for the input on the part of the user is not always of a kind where a question mark at the end is relevant – that is, it's not a question. Add to this the need to follow it with quotes – and then rub this out and type quotes again so you can use the cursor move keys to edit the line – just so a comma can be included where required in the input string, and straight away we aleady have two irrelevant characters on the screen before anything is actually typed in.

What's more, you always forget to use the opening quotes at the start, which itself suggests that it doesn't feel natural to have to do this. The situation is worse when inputting from an external channel – here not only the comma but also the colon and semi-colon are interpreted as carriage returns, or the end of the string, whereas in actual fact it may be nothing of the sort. Again, enclosing the string in quotes before it is saved to the external device prevents this, but on inputting it back the quotes will have disappeared. In either case, whether inputting from the keyboard or an external device, quotes cannot be included as part of the text. But why not, you may ask?

"GET" will input anything from the keyboard, and a string comprising all printable characters can be built up using "GET". But the Basic coding required to interactively edit this string at the same time rapidly becomes unwieldy and slow. Also, the input is not automatically echoed to the screen, so this must be done in Basic. There's no flashing cursor either.

A cursor can be forced, but it has the peculiar behaviour of leaving some characters in inverse video as soon as the cursor move keys are used. "GET" can be used to input text from an external device, but using Basic commands that retrieve one character at a time and build it into a string can be time-consuming to say the least. It's frustrating when all you want is input some text of an 'unusual' format.

If you're a machine code programmer of any skill, then this can be got around, except that you may find yourself having to re-develop the coding to suit the requirements of different programs. But if you want to stay in the Basic environment, then while 'GET" and "Input( )" have their place (and are very good at what they do), what is really needed is a new Basic keyword, one than will input strings of all printable characters.

Well here it is. Inspired, as it happens, by the IBM PC BASIC A's action keyword of the same name the utility is called "LINE INPUT( )" and is in the form of a machine code routine stored at $CAC0 to SCBBD high up in the free RAM area above the 64's BASIC interpreter. In actual fact, it is not a real keyword at all, but a "SYS" call which cheats by pretending to be an interpreter routine.

This is necessary because of the sort of information that the routine needs in order to carry out its task. The "USR" function only allows one numeric value to be passed on to a machine code routine, which then can only return one other value to a variable. The temporary register storages for "SYS" calls are rather long-winded to use, and in any case none of these are any use because "LINE INPUT( )" must return a string in a string variable.

The simple answer is for the routine, "LINE", to get these parameters direct from the current Basic line being processed using the subroutine "CHRGET" – which retrieves the next BASIC byte into the accumulator, and also incidentally is the thing that skips spaces not in quotes – in exactly the same way as all the interpreter routines get their information. This makes "LINE INPUT" very easy to program in Basic, because if you excuse the presence of the word "SYS" at the front it will appear in the listing, and in operation, like a valid Basic keyword. Furthermore, if you make a numeric variable equal to the "SYS" call address and name this variable "LINE", the subterfuge will be complete. The syntax then is as follows:

SYS < address .> INPUT (# < channel No.>,) < string variable >

Note the full stop between < address > and INPUT(#). This is very important as it ensures that the following valid BASIC keyword, "INPUT" or "INPUT#", is crunched down to its proper BASIC token when the BASIC line which contains it is entered. Neither of these will be executed by the interpreter in the normal way. However, they exist here exclusively for use by "LINE". In fact, the processing activity of "LINE" will make whichever of these two tokens inaccessible to the interpreter.

The keyword for "INPUT" or "INPUT#" is included because "LINE" has to vector within itself to the appropriate processing for either inputting from the keyboard and screen, or from an external device. The keyword is tokenised for three reasons: 1. to save memory space, although this will be negligible; 2. because it's a professional way of doing it; 3. since "LINE" uses it to find out which form of input is required, it simplifies things greatly if only one byte can be examined instead of a string of characters. This token, and the remaining parameters, are read by "LINE" direct from the Basic text using "CHRGET".

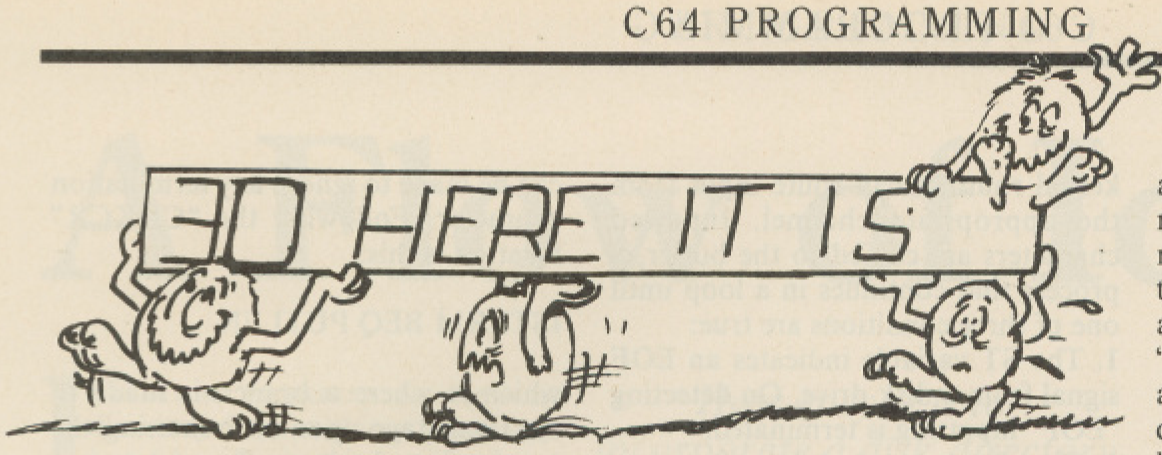
A number of subroutines are used by "LINE" to read the parameters. They themselves get the Basic bytes using "CHRGET". One of these is a contrived subroutine alled "GETVAL" located at $CBA7 (52135 decimal), and looks like this:

```
READ      EQU   $AD8A
FIX       EQU   $B7F7
ORG             $CBA7
GETVAL    JSR   READ
JMP             FIX
```

"READ" is part of a Basic interpreter routine which pickes up a numeric value from the Basic line currently being processed. The value may be written as a string of decimal numbers, or be in the form of a numerical variable. Whichever, "READ" converts it into floating point format and stores in Floating Point Accumulator # 1.

"FIX" is the familiar 'float to fix' interpreter routine. The contents of FAC 1 are converted into a 16-bit integer and stored in zero page locations $14-$15 hex.

Therefore "GETVAL" performs the action of reading a numeric value or a variable's contents from a Basic line and making same available in handy double-byte integer format in $14-$15 for a machine code to use, if "GETVAL" is handled properly. It helps in understanding how "LINE" operates, that the C64's Baisc interpreter follows a special convention that every interpreter routine, including those used by "GETVAL", expects the first Basic byte that it has to deal with to already be in the CPU's accumulator. Or, to put it another



way, every Basic interpreter routine calls "CHRGET" to get the next byte of Basic text into the accumulator before it exists. Also "GETVAL" is a little bit limited, only positive numbers in the range 0-65535 are allowable otherwise an "ILLEGAL QUANTITY" error is generated. Invalidly entered numbers produce "SYNTAX ERROR".

On calling "LINE" with the "SYS" command, true to the convention, the interpreter runs "CHRGET" to load the accumulator with the next byte of Basic text. Unfortunately this doesn't arrive intact when "LINE" takes control, because the action of executive "SYS" loads the accumulator with the contents of the 'temporary storage for A reg.' location at $030C (780 decimal). So the accumulator has to be refreshed with a called to "CHRGOT", a letter part of "CHRGET" which gets the same BASIC byte again. This byte must be a full stop (".") or $2E, if it is not a jump is made to $AF08 which prints "SYNTAX ERROR IN LINE <n>", and terminates the Basic program. The byte following this, retrieved by "CHRGET", must either be a token for "INPUT" or "INPUT #". If it is neither of these again a jump is made to "SYNTAX ERROR".

Otherwise at this point the routine "LINE" deviates. The bulk of the routine comprises two separate processes, the one chosen depends on whether the input is from the keyboard and screen, or whether it is from an external device.

LINE INPUT from keyboard and screen –

Syntax:
SYS < address>. INPUT < string variable >

Example:

```
100 LINE = 51904
110 SYS LINE.INPUT A$
```

Input from keyboard and screen is carried out in three stages. Firstly the screen RAM address for the start of the input is found by locating the cursor position with the kernal routine "PLOT", and calculating the screen address by the rather crude method of adding 40 to the screen base of 1024 'y' (row count) number of times. 'x' (column count) is added to the result. I did originally write a proper multiplication routine, but this required so much code that it wasn't really worth it. The calculated address is stored in free zero page locations as a pointer.

All you see on the screen is a flashing cursor with no "?" prompt. Now a kernal routine called "CHRIN" handles all the character getting and printing with full editing support and proper flashing cursor. In use it is identical to using the Basic editor in direct mode. It has the same limitations – if quotes are keyed then cursor move keys come onto the screen as characters, but rubbing out quotes and re-keying stops this. "LINE" discovers the length of the line. This is done by searching backwards from the maximum string length position, the default of which is 80, looking for a non-space character. On finding a non-space character, its position ahead of the start of input becomes the new string length.

The screen line, from the start address up to the string length, is copied to a buffer. This is our old friend the cassette header buffer at $033C. However $033C is reserved to hold the string length count, so the buffer actually starts at $033D. A conversion routine is used to translate screen coded characters into CBM ASCII, and the most significant bit is stripped off in case the screen characters were inverse video ("LINE INPUT" needn't input a line which has just been typed, it could just as easily have been PRINTed instead and the keyboard buffer POKEd with a carriage return).

At the end of this the string is stored in the buffer and its length in $03CC. At the final stage "LINE" calls interpreter routines "LOCATE", which goes in search of the string variable through the variable storage area – "LOCATE" gets the variable's name from the Basic line. If it does not exist then it is created. After this the string storage for the variable, if already in existence, is freed (by "FRESTR") and then a fresh storage area is defined using "STRRES" which does so according to the new string length passed to it. We now have a place in the string storage area to which the contents of the buffer can be copied.

It is impossible for characters to be put in this area directly (without being stored in a buffer first) because "LINE" does not yet know how long the string will be until this stage.

In this manner any string variable can be created or updated using "LINE.INPUT" in just the same way as it can by any other INPUT function. However some care is required where the variable is one of a dimensioned array. Small arrays up to a 'safe' limit of say 10 or 11 elements would be okay, in other words:

```
110 SYS LINE.INPUT T$(n)
```

Where 'n' is an index from 0 to 10 in the default provision for arrays up to 11 elements, but otherwise a 'working variable' such as IN$ for example would have to be used to transfer the input to the element where this is one of a large, dimensioned array, e.g.:

```
110 SYS LINE.INPUT IN$
120 A$(n) = IN$
```

If this precuation is not observed the process becomes too complex for the simpler than normal string building techniques used and usually crashes the 64.

Whatever the string is stored in the proper place, and the extra does not add perceptively to the processing time, even if a large string array is being filled with text.

Because "LINE" is located in RAM it can be modified. I mentioned that for input from keyboard and screen, it has a default maximum string length of 80 screen characters (two screen lines). It doesn't matter in which column across the screen the start of input occurs, the maximum remains 80 characters. You can change the maximum length by a POKE to suit your own requirements. For example, a disk file name cannot exceed 16 characters in length. If "LINE. INPUT" is used to get the file name it can first be modified only to accept

up to a maximum of 16 characters. In the source listing the maximum string length is labelled "MAXLEN" and is at $CB4D (52045). Use the following:

```
100 LINE = 51904
. . .
1000 MAXLEN = 52045
1010 POKE MAXLEN, 16
1020 PRINT "ENTER A NAME
FOR THIS FILE";
1030 SYS LINE.INPUT IN$
1040 NM$ = IN$+",W,S"
1050 OPEN 2,8,2, NM$
1060 etc.
```

You can type as many characters as you like for the file name, only the first 16 will be taken any notice of after you press [RETURN].

Be aware though of a couple of limitations of "LINE.INPUT". One is the same as that of the normal "INPUT", namely if other characters present on the screen come within the range of the maximum string length, it will be assumed that these constitute the end of the string, even if you don't want them. Enough blank space must exist beyond the position where input commences to prevent this.

Also, "LINE.INPUT" doesn't know if the start of input occurs on the bottom line of the screen. If the screen scrolls up where text exceeds one screen line, the start address is not adjusted up with it. Also "LINE.INPUT" will assume that the 'rubbish' beyond the top of the screen RAM area are characters to be input! Consequently "LINE.INPUT" should not be used any lower than the 24th row down.

LINE INPUT # from an external device –

Syntax:

SYS < address >. INPUT < channel No. > , < string variable

Example:

```
100 LINE = 51094
. . .
1000 OPEN2, 8, 2, "TEXT"
1010 SYS LINE.INPUT   2,A$
1020 etc.
```

This is the only time that "GETVAL" is used, and even then only the least significant byte of the resultant integer is needed!

"GETVAL" retrieves the channel number following "INPUT # " so that kernal routines can route input from the appropriate channel. Inputted characters are copied to the buffer, a process that continues in a loop until one of three conditions are true:

1. The ST variable indicates an EOF signal from a disk drive. On detecting "EOF" inputting is terminated.

2. The character read in is found to be the termination character. The default termination character is a carriage return (13). on encountering the termination character inputting is terminated. The termination character is NOT copied to the buffer.

3. The maximum number of bytes have been copied to the buffer. The default maximum number of bytes is 128. On the 128th character being copied to the buffer, inputting is terminated.

On input being terminated, "LINE.INPUT # " goes to the final stage of building the string to memory exactly as before. As far as "EOF" is concerned, it will be required that the BASIC program using "LINE.INPUT # " should also monitor the "ST" variable. Re-using "LINE.INPUT # " and forcing input past EOF causes a rubbish character to be copied to the buffer, overwriting the previous string. This is not a problem if other variables have been made equal to this string.

As before, because the routine exists in RAM, it can be customised to behave in a special way. "CHKCR" is a label in the source listing which represents the point where the character read is tested to see whether it's the termination character or not. It looks like:

```
CHKCR CMP # $0D
```

Where the operand is the value, in this case 13. This can be changed as in the following example:

```
CHKCR = 52065
POKE CHKCR + 1,0
```

Now the routine will stop reading when it encounters a zero byte. Similarly the maximum number of bytes copied to the buffer can be changed:

```
CHKMAX = 52079
POKE CHKMAX + 1,< n >
```

Where < n > is any number up to a limit of 191, because this is the maximum amount of space available in the cassette buffer area.

Even better, "LINE.INPUT # " can be made to ignore any termination character. Following the "CHKCR" location is this:

```
ISTERM BEQ PUTLEN
```

which is where a branch is made to the final stage upon encountering the termination character. By using:

```
ISTERM = 52067
POKE ISTERM, 234 : POKE
ISTERM + 1, 234
```

This has the effect of overwriting the branch instruction with "NOP'S", so the routine cannot exit if a termination character is found. This allows enormous flexibility for inputting data of a 'strange' nature – "LINE. INPUT " can read in all byte values 0-255. It is possible for example to read machine code into string – supposing you wanted to do such a thing!

Because "LINE INPUT(#)" is completely self contained, i.e. no part of the conventional interpreter INPUT(#) routines are employed, we are allowed to do something normally quite illegal. The following is possible in direct mode:

```
LINE = 51904

READY.
OPEN 2, 8, 2, "TEXT"

READY
SYS LINE.INPUT   2,A$

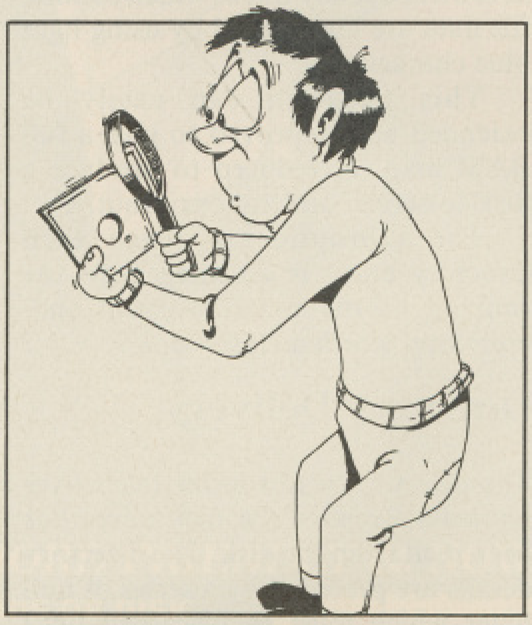
READY.
?A$
```

This is incredibly useful for verifying that a Basic routine that you are trying to de-bug has saved data properly or not. It's normally impossible to get at this data in direct mode, since using "INPUT # " results in an illegal direct command error. You can go on moving the cursor up an dinputting and displaying successive fields of data. This even works with numeric data saved using "PRINT # ", since these are written as decimal strings.

If you have an assembler you can enter the source listing, and if needs must the ORG statement can be changed to relocate the routine anywhere you like, but make a note of the new label locations if you then want to customise it with POKEs. It's unlikely that you will need to use "LINE INPUT(#)" very often, but when you do, you'll be glad it's there.

A Flow of Ideas

How easy is it to incorporate a disk directory reader in a program?
By Norman Doyle



It's often necessary to view a directory from within a program, but from Basic this can be very difficult. There is a way, and this method can reveal more information than may at first be obvious.

Before a disk directory program can be written, it is essential to know what data the drive makes available. Using the following short program on the directory in Table 1 gives all of the information which is displayed in Table 2.

```
10 OPEN 8,8,1,"$,S,R"

20 GET#8,A$:IFA$=""THENA$=CHR$(0)

30 POKE49152+A,ASC(A$):A=A+1

40 IFA=0THEN20

50 CLOSE8
```

The rendered data includes the Block Allocation Map (BAM) which lies from $C002-$CXXX, as well as the directory content itself ($CXXX-$CXXX). Using GET # , each byte can be read and used to form quite a powerful source of disk information.

From this readout, the aims of the program can be derived. What we will create is a flow diagram for a program which lists each directory entry, the file type, blocks used and the track and sector (the specific block location) relating to each file. As the program runs, it will keep track of the total number of programs on the disk for a general round-up screen, which is to be displayed after the individual program detail screens.

The program can also keep a tally of the total number of blocks used, and this value can then be compared with the number derived from the BAM to ensure that the disk has been validated correctly. Finally, a detailed map of the number of free sectors (blocks) on each track can be displayed using colour to differentiate between tracks which have not been used at all, and those which have lost a few sectors to file storage.

The first duty of the program is to set the screen colours and then initialise the disk under examination. This means opening command channel 15, and keeping it open for checks on disk errors throughout the program.

Next, the directory file is opened for a sequential read operation. After the drive loads the first sector into its internal buffer, variables are initialised and dimensioned ready for the main program to begin. The reading of the BAM takes a little time, so a comforting message is displayed to assure the user that this is a normal occurrence.

Exploded BAM

The layout of the BAM can be seen from Table 3. There is no single byte pair responsible for recording the number of free bytes on the disk, so this value has to be derived from the data at the beginning of the BAM. The first two bytes (the track and sector links) are not loaded when the directory is read as a sequential file, and the next two bytes are irrelevant, but some of the next group of bytes are essential to calculating the number of free memory blocks on the disk.

The information is stored in a specific way, with the bytes grouped into clusters of four. The first byte is the total number of free blocks on a particular track, and the three bytes that follow can be used to calculate which particular sectors these are. In this case, the only byte of interest is the total number of free blocks, so that is read and added to a running total while the others are discarded.

Apart from keeping a record of the total number of free blocks, each track's details must be stored in an array if the aim of producing a tracks map is to be fulfilled. There are 35 tracks on a normal Commodore disk, and it can be seen from Table 2 that there are, in fact, 35 groups of four bytes in the BAM. A loop must read each of the groups, storing the first byte in an array and adding its value to a grand total of free blocks. After this is done, the next three bytes can be ignored.

At the end of the BAM, the disk name and ID numbers can be found. This must be read and stored in a suitable form for display at the head of each screen page. First of all, the string is initialised to produce a label indicating that what follows is the disk's title and, to comply with Commodore's convention, the RVS ON character is added to display the disk header in reversed characters. A second convention is to place the disk name in quotes. This is not catered for from reading the directory file, so the opening quote is added at this point.

The disk name is allocated a space of 16 characters in the BAM sector of the directory. This means that by simply reading the 16 character group and adding it to the title string, any disk name can be catered for without any complicated checks. The resultant string is then completed with a closing quote mark.

To complete the screen page header, the next 15 bytes are added to the string. This actually reads more than enough characters to cater for the ID and the disk type descriptor (always 2a) on a normally formatted disk, but is extended to cater for non-standard formatting using four character IDs and other such tricks. The difficulty here is that some of the bytes will be stored as 'null values', which is the reason why string variables are being used in the program instead of numerical ones to avoid generating errors.

Because the bytes are stored as ASCII values, the conversion routine must look for these nulls and convert them to CHR$(0) to avoid similar errors when the bytes are converted for use. This happens quite a lot, and a subroutine is the ideal answer. Once the disk name has been rendered as

a string, it is printed and, because this string contains the clear screen symbol, the name acts as a title for the screen page.

Before reading in the directory entries, the column heads must be displayed. The convention that has been derived for this routine is that the program name will be followed by the file type, block count and the track and sector values for the first program block.

This information is not stored in the correct order so it must be read, stored and sorted out in a print statement. The layout of a single directory entry can be seen in Table 4. A subroutine deals with the actual reading of the individual entries, and this will be examined later.

As each directory entry is dealt with, a counter is incremented so that a check can be made for a full screen page (20 entries). When this number is reached, a 'Press Any Key' message is printed and a keypress detection loop initiated.

Directories come in all shapes and sizes, so the program cannot work on a simple loop to read in the entries. A way is needed to indicate when the directory is complete. This is done by reading the system variable, ST. When this has a value of zero, work is still in progress, but if it has a real value one of two things has occurred.

It may just be that the directory has been read in completely, or it could be that an error has occurred. Before closing the file, the routine checks the error channel. If an error is detected, the program will halt and the message is printed; if there is no error the program continues to the next stage.

Under a screen page title of 'Vital Statistics', the details of the number of programs, and blocks used are printed, suitably labelled. Both are derived from the individual directory entries and supplied via the 'read entry' subroutine.

The free blocks' value has been calculated from the BAM record, but before this is printed we can use this value and the 'blocks used' value to check for a correctly validated disk. If the number of blocks used is subtracted from the total number of blocks free on a newly formatted disk (664), the result should equal the BAM-derived free block count.

Any inequality means that the BAM is faulty, and this is flagged by a suitable message. A word of warning – inequalities could be the result of REL or USR files being used on the disk, so ensure that this is not the case before validating, or you could lose valuable data!

After the free blocks have been displayed, a table is drawn up with the tracks listed above their unused sector values. At each stage, these values are checked against the sector capacities for each track. If none of the sectors have been used, the value is printed in brown, but any tracks which contain file data are highlighted by using light blue characters.

This program can easily be extended and improved to give a full BAM map, or reduced to produce a disk-to-screen directory printout.

The subroutine to reach each directory entry is an integral to the running of the main program and, therefore, worth a closer look.

Directory Delivery

The directory read routine takes over immediately after the disk name has been read from the disk. The directory details are preceded by a series of null bytes which must be discarded until a 'real' byte value is read in. This method saves time but has one serious drawback when the first file has been deleted and not replaced.

File types are denoted by byte values as follows:

129 = SEQ
130 = PRG
131 = USR
132 = REL

If the file is 'locked', or protected, these values are increased by 64 to give a range of 193-196. Deleting a file simply results in these values being replaced by a null byte. Using the method of discarding null bytes would mean that the first significant value would be the old track number when a deleted file was encountered.

Consequently, the program includes a check to see if the value read is in excess of 35. If this is not true then it must be a deleted file, so 'DEL' is assigned as the program type and the routine jumps to evaluating the read in value as a track number. If the value is valid for a file type, a check is made to test if it is locked or not. Provision is made to denote a locked file by using a reversed 'less than' symbol, which is saved as a string. If the file is unprotected, this string remains as a space.

A second string is derived from the file descriptor value. This is the file type which is calculated directly from the file type value in conjunction with a MID$ statement. After the next byte has been read in, the pathway which the DEL file option took converges with the main program so that the track value can be assigned to a string before the sector value is read and similarly stored.

The disk name had 16 bytes assigned to it, and the same is true for the program name which follows the sector byte. A loop reads this in and concatenates a string. If a program name has less than 16 characters, the entry is padded out with shifted space values (160), and this is useful for formatting the screen printout later, so they are not discarded here.

After the name there are nine zero bytes which are read in and disregarded, so the next significant bytes denote the number of blocks that the program occupies. This value is stored in low byte/high byte format and a suitable routine is included to reveal the true decimal value. All that remains is to print the information out in the correct screen columns. Deleted files names are shown in orange and existing files in white.

One problem with filenames is that they sometimes contain control characters such as screen clear, case switch or even colours. To avoid these messing up the beautifully formatted display, location 212 is poked with a value to fool the computer into thinking that it is in quotes mode, so that the reversed character is printed instead of the action which it represents being executed. Once the rest of the information is printed, control is then returned to the main program.

Directory Flow

The flowcharts show the logic of the program contained in the Listings pages. This is rendered as a Basic routine, but the beauty of flowcharting is that the same logic pathways can be applied to machine code or just about any other language that has been invented.

If you decide to investigate directory reading further, you'll also soon realise how useful the charts are for modifying the program. It's far better than muddling through reams of listings, noting variables and then trying to make sense of it all.

THE ULTIMATE BACKUP CARTRIDGE GOES ONE BETTER...

NOW ACTION REPLAY Mk V PROFESSIONAL

FOR CBM64/128 HAS ARRIVED



EVEN MORE POWERFUL, MORE FRIENDLY & NOW EVEN MORE FEATURES!

ALL FEATURES AVAILABLE TO TAPE OR DISK USERS.

ACTION REPLAY Mk V differs from all other cartridges because it combines an 8K RAM with a FULL 32K operating system ROM. That means that ALL features are INSTANTLY AVAILABLE AT ALL TIMES.

WARNING!! Other systems use outdated technology which severly limits performance. Action Replay's state of the art hardware gives you MORE POWER, MORE SPEED, MORE FACILITIES than any other cartridge. There really is no comparison. Here are just some of the features...

✓ **TURBO RELOAD** Action Replay Mk V has 2 unique Turbo Loaders - "Ramloader" & "Warp 25". Both work at up to 25 times normal speed! ✓ **PRINTER DUMP** freeze any game & print out the screen. ✓ **PICTURE SAVE** save any HiRes multi colour screen to disk at the push of a button. ✓ **SPRITE CONTROL** full range of controls with 64K operation. ✓ **POKEFINDER GENERAL** an automatic infinite lives generator!
✓ **SUPER COMPACTOR** ultra efficient program compaction techniques. ✓ **TEXT SCREEN EDITOR** modify the text screen on a frozen program. ✓ **SUPERFAST DISK OPERATION** load 200 blocks in just SIX SECONDS. ✓ **MORE TAPE FACILITIES** you don't need a disk drive to use Action Replay. ✓ **TOOLKIT COMMANDS** full range all at the press of a button. ✓ **FULLY INTEGRATED OPERATION** with onboard custom LSI LOGIC PROCESSING CHIP.
✓ **CENTRONICS INTERFACE** for parallel printers. ✓ **PROFESSIONAL MACHINE CODE MONITOR** Full 64K monitor available at all times with full range of commands.

PERFORMANCE PROMISE

Action Replay has an unmatched range of onboard features. When you buy Action Replay, if you don't find our claims to be true, then return it within 14 days for a full refund.

GRAPHICS SUPPORT UTILITIES DISK

SLIDE SHOW. View your favourite screens in a slide show type display.

BLOW UP. A unique utility to allow you to take any part of of your picture & 'blow it up 'to full screen size.

SPRITE EDITOR. A complete sprite editor helps you to create or edit sprites.

MESSAGE MAKER. Takes your favourite screen created with a graphics package or captured with Action Replay & turns it into a scrolling screen message complete with music.

ONLY £12.99

NOTICE 1988 COPYRIGHT ACT

DATEL ELECTRONICS Ltd. neither authorizes or condones the use of it's products to reproduce copyright material. It is illegal to make copies of such material without the expressed consent of the copyright owners or thier licencees.

TOTAL BACKUP POWER

The most powerfull backup system ever devised. Unstoppable freezer system. Freeze at any point. Just press the magic button to backup your programs.
❑ TAPE TO TAPE ❑ TAPE TO DISK ❑
❑ DISK TO DISK ❑ DISK TO TAPE ❑
VERY fast & efficient program compaction. Single part save at TURBO speed to disk or tape. Backups turboload INDEPENDENTLY of cartridge. No "programing" or user knowledge required.
THE PROCESS IS AUTOMATIC - JUST GIVE THE BACKUP A NAME.

WHAT THE REVIEWERS SAID

"I'm stunned, amazed and totally impressed. This is easily the best value for money cartridge. *The Cartridge King!*"
Commodore Disk User

UPGRADE INFORMATION

Mk IV Professional to Mk V Professional - just send £9.99 & we will send you a new Mk V Operating System Chip for you to just plug in! Mk IV (Standard) to Mk V Professional - send your old cartridge plus £15.99 & we will upgrade it (allow 14 days).

ALL ORDERS NORMALLY DESPATCHED WITHIN 48 HRS

HOW TO ORDER ...

BY PHONE	BY POST	FAX
0782 744707 24hr Credit Card Line	Send cheques/POs made payable to "Datel Electronics"	0782 744292 UK ORDERS POST FREE EUROPE ADD £1 OVERSEAS ADD £3

PRICES AND SPECIFICATIONS CORRECT AT TIME OF PRESS AND SUBJECT TO CHANGE WITHOUT NOTICE

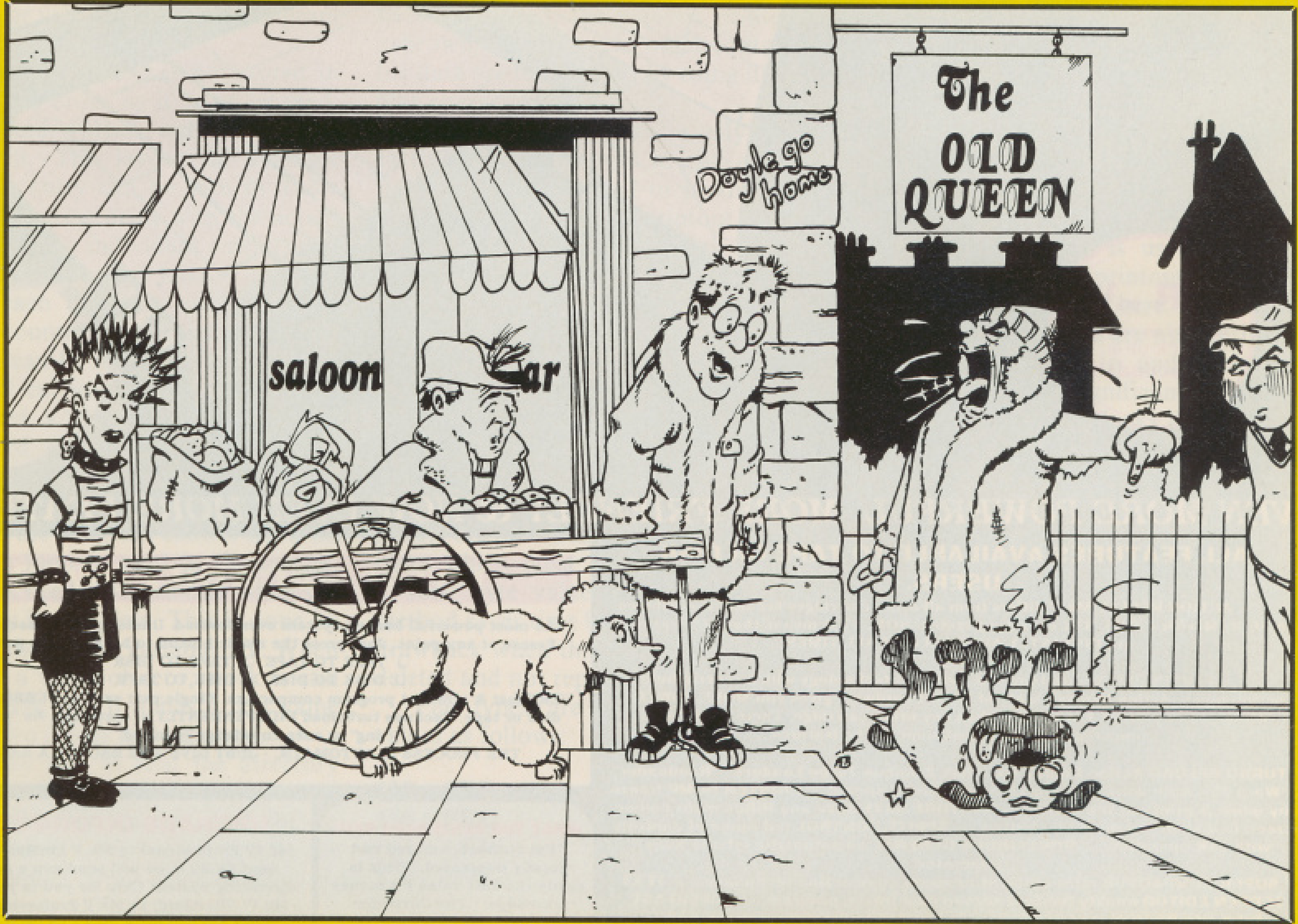
CALLERS WELCOME - Please reserve goods by telephone prior to visit.

DATEL ELECTRONICS

DATEL ELECTRONICS LTD., FENTON INDUSTRIAL ESTATE GOVAN ROAD, FENTON, STOKE-ON-TRENT, ENGLAND.

SALES ONLY	TECHNICAL ONLY
0782 744707	0782 744324

Deadenders



War in Middle Earth

The world of Tolkien's Middle Earth once again comes to the computer screen only this time as a wargame. In it, you control the forces of the Fellowship of the Ring in a desperate fight against the evil forces of Sauron. Your objective isn't military victory, it's far more important than that, as you must clear a path through the evil forces for a hobbit to take a Ring to the cracks of Doom.

At your disposal you have the dwarves of the Iron Hills, princely elves and men of good heart as well as the Fellowship party of Frodo, Gandalf and Aragorn. Frodo is, of course, the ringbearer and can wear it at will to escape the battlefield and the orcs. However, it has its dangers as wearing it corrupts the soul and takes you into the realm of the Nazguls who will attack you if they are near.

The main display shows the map of Middle Earth and a finger pointer with which you can zoom in on any area and view the forces in it and issue orders for them to move into battle. Eventually, forces will clash with the undisplayed

GAMES UPDATE

Deadenders is an adventure based on everyone's favourite soap opera – *Neighbours*. Nah, seriously guv, it all takes place in Herbert Square, home to the Squeals, Foulups and other unsavoury characters.

The story starts after Effl has lost her Willie. Yes, in case you haven't guessed, it's a spoof full of caricatures and one or two slightly naughty jokes.

You play the part of PC Donald Dance and your mission, should you choose to accept it, is to find out who did the dirty on Willie. The inlay tells you that poor old Willie turns up squashed, and with two holes in his neck. Leave it out, mate, the missis'll never believe this'un. It also goes on to say that you can talk to the characters that you meet. "Ask Bendy about Colon" is cited as an example. I tried it, and was met with a "You speak, but it doesn't further your enquiries." None of the Deadenders wanted to talk to me. Was it something I said?

The jokes are obviously geared towards Eastenders fans, but I can quite believe that non-followers of the series would get at least a smile from the game. Most of the humour, for me at least, came whenever I examined a character. When I examined Scary, I was told "Purple eyes and bright green hair, she resembles a technicolour skunk." Great stuff. All the descriptions are along these lines, and most are less than charitable.

When it comes to gameplay, I have a few reservations. The game doesn't accept some of the normal abbreviations. I had to type in "inventory" as opposed to just "i". Not all the exits were listed, even the ones that should have been obvious, and mapping left something to be desired. When standing on the landing of the Old Queen, I was faced with the kitchen, bedroom and living room, yet no directions were given. It was a case of "enter kitchen" etc, and no matter which of them I went into, the exit was always to the south. This system wasn't used consistently, though. From Herbert Square I could see the pub, the cafe and the corner shop. This time I had to input the usual N, S, E and W, and hope I went to the building I wanted. Some things should be taken for granted in an adventure and sticking to one way of inputting directions is on this agenda.

During the course of the game, I came across two bugs, one funny and one annoying. I'd got a box from Arfer and, whenever I tried to examine it, the game crashed. However, I was allowed to "search box" with no ill effects. Later, I discovered that I could "exam box" but only if I wasn't carrying it. I can just imagine how frustrating it would be for someone to find that out after going for a long time with no saves! The other bug was a silly one. I'd been searching through a dustbin, as one does, and my investigations took me to the launderette where I was reminded how disgusting my uniform was. No problem. I took it off, washed it in a machine, got it out again, then found out I had two uniforms, one wet and one dirty.

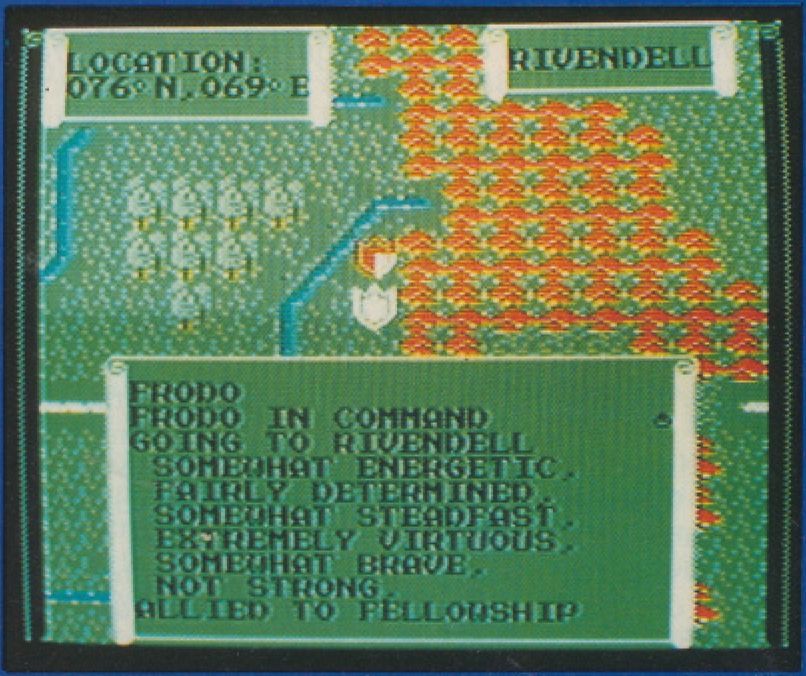
In keeping with the spirit of the game, there's no ordinary reply when you tell the computer to get something. Instead you get a chirpy "Right, me old china." Should any copies of this game reach America, it will only reinforce their beliefs that we all talk like Dick Van Dyke.

The authors have got a nice line in schoolboy humour, which I greatly appreciate. What a shame about the programming.

If you can ignore the niggles that I picked on, get down the frog an' toad an' give your dosh to the geezer in the shop. But if you're the sort who likes to be able to use single key inputs, logical geography and no bugs, knock it on the 'ead.

One last question. Where was Kylie?

Touchline:
Title: *Deadenders.* ***Supplier:*** *Top Ten Hits, MIL, 12 Chiltern Enterprise Centre, Station Road, Theale, Berks RG7 4AA.* ***Tel:*** *(0734) 302 600.* ***Machine:*** *C64.*



enemy and the screen will swap to the battle screen showing all the combatants, who will start the fight without your help.

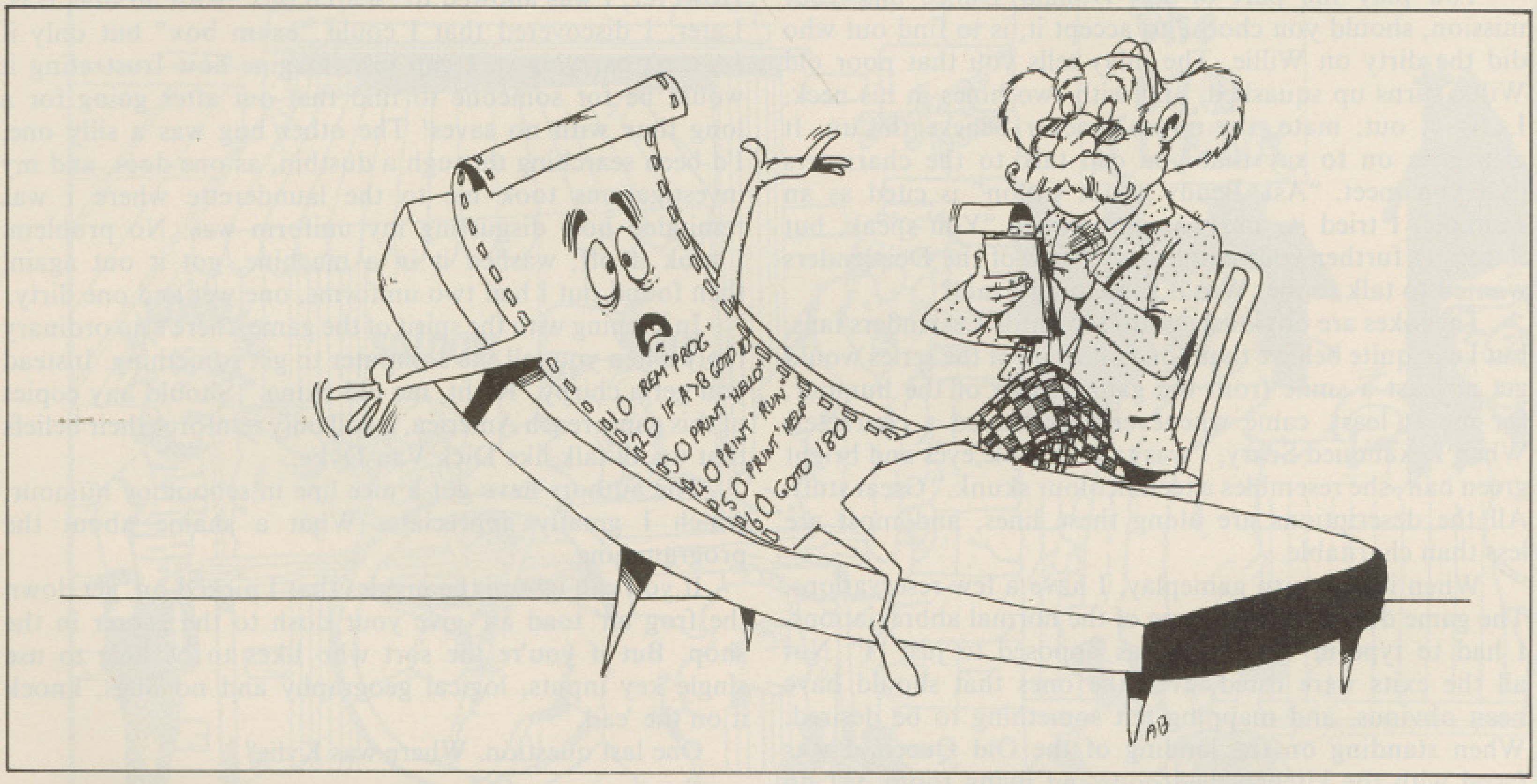
You can effect the battle by using a cursor to move men into the fray or you can control one character directly. This may sound like a good idea but it doesn't quite work, as to select a character you have to position the cursor over his feet and the battles just take too long. This can be annoying, particularly if the battle was only a decoy action while the party move through the area.

Unfortunately, this spoilt an otherwise interesting game which could have been so good.

Touchline:
Title: *War in Middle Earth.* ***Supplier:*** *Melbourne House, 2-4 Vernon Yard, Portobello Rd, London W11 2DX.* ***Tel:*** *01-727 8070.* ***Machine:*** *C64/128* ***Price:*** *£14.99 (Disk) £9.99 (Cass).*

Program Analysis

Programming can be made simple with these three C64 performance analysers



COMMODORE 64 PERFORMANCE ANALYSER

Basic is a programming language which makes it very easy for programmers to create complex programs with a minimum of effort. We pay a price for this programming ease, and that price is often poor performance, that is, our Basic program runs slowly. Another problem which confronts the Basic programmer is what to do when a program runs without failing, but doesn't give you the results you expect. How do you find out what your program is doing without adding PRINT statements to your program to trace execution or interrupt execution at strategic points?

The Performance Analyser helps to overcome both problems. Not only does it trace the logic flow in a Basic program, it also determines how long each Basic line took to execute. Thus the Performance Analyser is a generalised performance analysis tool for the Commodore 64.

Performance Analyser TRACE Facility

Most commercial traces usually amount to a window displaying five or six line numbers on the screen as your Basic program runs. The line numbers scroll in the window as each line is executed, and the window may or may not interfere with your program output. You normally cannot trace a Basic program which uses hi-res graphics, and you certainly cannot go back and check the line number sequence previously displayed. Although you can usually slow the trace display down (by the space bar for example), you have very little chance of writing down the line numbers on paper for a more detailed analysis.

The Performance Analyser overcomes all of these problems. It allows you to trace any Basic program which uses normal screen graphics, hi-res screens, sprites or sound and does not interfere with the operation of the program. The Analyser will not slow your program down, and allows you to give the trace display at your leisure. You may scroll backwards or forwards through the line numbers for as long as you wish.

Performance Analysis

The Analyser also provides you with a tool to determine how efficient your Basic program is. When it displays the line number it also displays the time it took to execute the line. As you scroll through the line numbers you can tell at a glance which line numbers are slowing down execution and which line numbers are executed most often. Basic programs are the same as any other program – they follow the 80/20 rule. That is, 80 per cent of the work is usually done by 20 per cent of the program. The Performance Analyser is the tool you need to tell you which 20 per cent of your program is doing 80 per cent of the work, and how long it is taking to do it. You can then concentrate on making that part of your program more efficient.

Analysing a Basic program

The Analyser is written entirely in Machine Language, and is designed to cause as little interference as possible with the traced program. The Analyser is normally loaded at 38912, and all Analyser variables and constants are contained in the 2k from 38192 to 40959. Your Basic program thus has the RAM between 2048 and 38911, any low storage locations it requires and the free RAM at 49152. Should you require the RAM at 38912, then set the top of Basic pointer (55,56) to the last RAM location available to Basic, and the Analyser will use 2K of RAM before this address. For example, if the top of the basic pointer is set to 32768, then the Analyser will load itself at 30720.

Type in the Analyser loader program and save it as ANALYSER1. Make sure you verify that what you saved is correct. To use the Analyser, simply issue a load "ANALYSER1" after setting the top of BASIC pointer if necessary. ANALYSER1 will set the required Basic pointers, POKE the Analyser Machine Language logic into the correct RAM, relocate all required ML addresses and print messages to indicate how to start and stop the Analyser and display the trace data. The following messages are displayed on the screen by ANALYSER1 during execution;

```
LOADING THE ANALYSER AT
38912
LOAD OK
RELOCATION OK
1. START ANALYSER = SYS 38912
2. STOP ANALYSER = SYS 38915
3. DISPLAY DATA = SYS 38918
```

If the load fails, or the relocation of addresses fails, a message is issued and ANALSYER1 stops.

Obviously to start the Performance Analyser you SYS to 38912 or to the address displayed by ANALYSER1. You can do this from a program or from direct mode. The message TRACE STARTED is displayed by the Analyser, unless you start it from a program. The message is not issued then to ensure that the Analyser does not interfere with program messages or displays.

After the Analyser has been loaded, you then LOAD the Basic program or programs you wish to analyse. The Analyser monitors execution of your program(s), and saves trace data in the trace data buffer for later display. If you only want to trace part of a Basic program, you would do the following;

```
1000 REM START THE
ANALYSER
1010 SYS 38912
1020 FORTH = .6to6STEP.1
1030 X = A*COS(TH)
1040 Y =B*COS(TH)/C
1050 NEXT
1060 REM STOP THE ANALYSER
1070 SYS 38915
1080 REM DISPLAY TRACE
DATA
1090 SYS 38918
1100 END
```

After your Basic program has finished, or you stop it executing, you can stop the Analyser if you want to. However, you don't stop it to display the trace data. You may leave it active to trace another program if you want to.

Obviously, to stop the Performance Analyser you SYS to 38915 or to the address displayed by ANALYSER1. You can do this from a program or from direct mode. The message TRACE STOPPED is displayed by the Analyser, unless you stop it from a program. Again the message is not issued to ensure that the Analyser does not interfere with program messages or displays.

Finally, you may display trace data at any time by entering SYS 38918 or SYS to the address displayed by ANALYSER1, and of course you may do this in direct mode or from a program. The message NO TRACE DATA is displayed by the Analyser if there is nothing to display. Again the message is not issued if you are under program control. This is to ensure that the Analyser does not interfere with program messages or displays.

If there is data to display the Analyser presents it in full-screen mode, that is a page or full screen data consisting of line numbers and line execution times is displayed and the Analyser ML program waits for you to press one of the function keys; F1 terminates the display, F5 scrolls back to the previous page of data and F7 scrolls forward to the next page of data.

You may scroll back and forward through the trace data for as long as you like with function keys F5 and F7. When the end of the trace data is found, the number of lines executed and the total execution time is displayed, and the Analyser ML program waits for you to press a function key. The Analyser will only recognise F1, F5 and F7 function keys. All other keys are ignored. If you scroll forward from the end of the display, you wrap around to the start of the trace data again. You can't scroll back from the top of the trace data, you may only scroll forward.

NOTE: Trace data will be displayed automatically when the trace data buffer area is full. The trace data buffer is actually the RAM under the BASIC ROM. As much trace data as possible is stored there before the execution of the Basic program is interrupted and the trace data displayed. If you want your Basic program to continue, simply press F1 and the trace display is terminated. Your program begins execution from where it was interrupted. If you want to browse the trace data, then use F5 or F7 to scroll back and forward through the data.

How the Performance Analyser Works

The Analyser works by monitoring the execution of Basic programs via the character dispatch vector in low storage. As each program byte is interpreted, the Analyser checks to determine if the current line number (57,58) has changed from the previous byte read. When the line number changes, then the Analyser stores the line number and current time in the trace data buffer under the BASIC ROM. This is done until such time as the trace data buffer is full.

When the buffer is full, the Analyser saves the first 2K of low storage (0-2047), colour RAM and various control registers in the RAM under the KERNAL ROM. The trace data is then displayed, and when the display is stopped via function key F1, the Analyser restores the first 2K of low storage, the colour RAM and the various control registers. This allows the Basic program to restart execution from the point where it was interrupted, and the program screen is restored, as well as character colours and backgrounds.

If your Basic program uses the RAM under the Basic of KERNAL ROMS, then you cannot analyse it with this utility. Note also that if your

Basic program resets the time (TI$ = "000000"), then the Analyser will not fail, but the execution times displayed will be unpredictable.

COMMODORE 64 PROGRAM ANALYSIS

Commodore 64 Program Analysis (C64PANAL) is a Basic program which analyses the contents of any Basic program and displays the information on the screen or printer. C64PANAL first displays summary information which contains the program name, the size of the program in bytes, the number of lines in the program, the total number of commands (ie PRINT's, GOTO's, If's etc.) and the number of variables.

Once the summary data has been viewed, a detailed list of the commands used in the Basic program and the number of times each command is used is displayed. When you have finished viewing the command data, a detailed list of the variables and the use of each variable is displayed, and when you have finished viewing the variable data you may end the display, ask for the information to be reshown or send the data to your printer.

Using C64PANAL

C64PANAL allows you to analyse your Basic program. It does this by running in the 4K of free RAM at 49152 to 53247, and loading the Basic programs it analyses at 2049. By not using the RAM between 2048 and 40960, C64PANAL is capable of analysing the largest Basic program. However, with only 4K of RAM to run in, C64PANAL will run slowly analysing large Basic programs because many garbage collections will be done to ensure that there is sufficient space for C64PANAL to operate correctly. Also, only 50 variables can be displayed because of space constraints.

Obviously if C64PANAL is to run in the RAM at 49152 then some changes need to be made to Basic pointers in low storage. The start of Basic and end of Basic addresses need to be changed as well as the start of variables etc. These changes are handled by the C64PANAL loader program. LOADER is the Basic loader program which automatically loads C64PANAL. It sets the low storage pointers, and then uses the dynamic key facility to automatically load C64PANAL.

You must create and save LOADER first on tape or disk. Next type in C64PANAL and save it directly after LOADER on tape or on the same disk as C64PANAL.

Note that if you are using disk you need to change line no 10 in LOADER from LOAD "C64PANAL", 1,1 to LOAD"C64PANAL", 8,1 so that C64PANAL will be loaded from disk and not tape.

Once you have saved LOADER and C64PANAL to tape or disk then simply load LOADER and RUN it. LOADER will set the various low storage pointers and then set up the screen and keyboard buffer so that when it ends, C64PANAL is automatically loaded at 49152. When C64PANAL has been loaded it begins execution automatically, clears the screen and places the first message on the screen:

LOAD FROM DISK (Y/N)?

If you want C64PANAL to load the Basic program it analyses from disk, then reply Y. Otherwise reply N and the program will be loaded from tape. Before replying to this message, you should have the tape or disk which contains the program to be analysed in the datasette or disk drive.

The next message to be displayed is:

PROGRAM TO BE LOADED = = ?

Your answer to this message tells C64PANAL the name of the program it is to load from tape or disk to analyse.

C64PANAL then uses the KERNAL load subroutine to load the Basic program at 2049 and begins to analyse it. Since it may take some time to analyse large Basic programs, C64PANAL places the line number being analysed in the top left-hand corner of the screen while scanning the Basic program. When analysing is finished the summary report is displayed as follows:

```
----PROGRAM STATISTICS----
PROGRAM NAME = C64PANAL
PROGRAM SIZE = xxxxx
NO OF LINES = xxxxx
NO OF COMMANDS = xxxxx
NO OF VARIABLES = xxx
USE ANY KEY TO CONTINUE
```

You may view the summary report for as long as you wish. To move to the command report, simply use any key and the following display appears on screen:

```
- COMMANDS -
END        =1
FOR        = 5
NEXT       = 6
DATA       =10
INPUT      =1
READ       =1
GOTO       =25
IF         =30
GOSUB      =17
RETURN     = 17
REM        =8
POKE       =5
PRINT      = 38
THEN       =25
+          = 45
-          = 15
*          = 51
/          = 19
AND        = 1
=          = 55
MID$       = 3

USE ANY KEY TO CONTINUE
```

If all commands used in the program can be displayed on one screen, then when you press any key you will move to the VARIABLE display. If more commands are used than can be displayed on one screen, then the next screen of data will contain command data. When the last of the command data has been displayed and the USE ANY KEY message is displayed, when you press any key the list of variables appears on the screen. Note that + - * / = < and > are considered commands when used in statements such as A=A+B*C-D/E of IF X 0 GOTO 1000.

When the commands are finished, the list of variables is displayed as shown

```
- VARIABLES -
I          = 2
X          = 4
RR$        = 5
Z          = 3
ZZ%        = 9
USE ANY KEY TO CONTINUE
```

When the last of thevariables has been listed, C64PANAL displays the following message

R = RE-DISPLAY, X = END, P = PRINTER

If you press the R key, then all information beginning with the summary display is reshown. If you press the X key then program execution is terminated and the final time message is displayed:

TIME TAKEN = xxxx.xx

This is the time in seconds it has taken C64PANAL to analyse your program. You may then use C64PANAL to analyse another Basic program.

Press P and the information is sent to the printer.

Applying C64PANAL

C64PANAL has many uses. You can find the size of your Basic program, the number of variables you use and the number of lines in your program. The number of lines is important, because each line in a Basic program carries an overhead of 4 bytes (2 bytes for a link address and 2 bytes for the line number). A 500 line program uses 2,000 bytes of storage for link addresses and line numbers. If you have an excessive number of lines, you can conserve space by reducing the number of lines (also known as crunching your program). You reduce the number of lines by placing multiple commands on the same line seperated by colons, removing blanks and removing REM commands.

Processing new lines also carries with it a performance penalty. The more lines in a Basic program, the longer it normally takes to run. By reducing the number of lines, you normally reduce program execution time. C64PANAL will tell you how successful you have been at reducing the number of lines in your program. It will give the size of your program and the number of lines before crunching, and then after you have made your changes you can run it again and get the new figures.

The detailed list of commands (ie PRINT's, GOTO's, IF's etc.) can also be used to reduce program size and increase performance. For instance, if you find that you have a very large number of IF commands, then you may be able to reduce them by using the ON command. For example if you have:

```
IF CC = 1 GOTO 1000
IF CC = 2 GOTO 1100
IF CC = 3 GOTO 1200
IF CC = 4 GOTO 2000
IF CC = 5 GOTO 2100
IF CC = 6 GOTO 2200
```

then you could replace the IF commands with one:

```
ON CC GOTO 1000, 1100, 1200, 2000, 2100, 2200
```

It is also interesting to see the pattern of commands in various programs and which commands are used most frequently. In string operations the LEFT$, RIGHT$, MID$ etc. will figure prominently. However, the most common commands used are the IF, GOTO, FOR and NEXT and PRINT.

The list of variables is a powerful tool to help in the exeuction of your Basic program. Basic maintains a list of variables, and the closer a variable is to the start of that list, the less time that is needed to find the variable when it is referenced in a statement. For example, every time IF X = 3.7 TOTO 100 is executed, the X variable must be found in Basic's list of variables to check if it is 3.7 or not. Thus the closer X is to the top of the list, the faster it is found. The order of variables makes a significant difference to the execution time of your program if you have a large number of them. C64PANAL helps by giving you a guide as to which variables ought to be defined first so that they appear at the top of Basic's list of variables. You can ensure the order of variables by defining them in the following anner:

```
X = 0:A = 0:Y = 0:PC = 0:TK% = 0 etc.
```

X will come first, A second, Y third in the list and so on.

If you have Basic programs where execution time is crucial (for eg. games programs) then C64PANAL will be an important tool to help you analyse those programs and make them faster.

COMMODORE 64 SWITCH

Commodore 64 SWITCH is a short Machine Language (ML) program which resides in RAM just before the BASIC ROM. It occupies storage locations 40704 to 40959. C64 SWITCH allows you to partition your C64 into two logical machines. You switch between the two partitions or regions with a single key stroke. With this utility, you can load two Basic programs at once and compare them or work on them. However, you cannot have both programs running simultaneously.

Using C64 SWITCH

C64 SWITCH allows you to set variable region sizes. The regions are designated zero (0) and one (1) and region 0 will extend from location 2049 to the limit your set, while region 1 extends from the end of region 0 to 40703.

To use the switching function, simply load SWITCH, which is a Basic loader program. When you run it, SWITCH will load the ML routine at 40704 and display the message:

ENTER REGION 0 ENDING ADDR = = >?

You enter the ending address for region 0 (and thus region 1 starting addr) and the final messages are displayed:

REGIONS 0 and 1 INITIALISED
REGION ACTIVATED = 0

To switch between the two regions use the F1/F7 keys. F1 will activate region 0 while F7 will activate region 1. The active region is displayed in the upper right-hand corner of the screen in reverse video. To deactivate the SWITCH, simply hit RUN STOP/RESTORE or turn the C64 off and on.

Applying SWITCH

C64 SWITCH has three main uses. You can load two Basic programs at once, and work on them or compare them. You can use region 1 as a data region which is accessed by a program in region 0 (SWITCH was originally written for this purpose). Finally, you can use SWITCH as a means of merging two programs. If you want to add code to a program in region 0 from a program in region 1, simply LIST the statements in region 1 on the screen, press F1 to activate region 0 and then move the cursor over the lines you want added and press RETURN. Each line will be entered into the program in region 0.

Help!

Enhance your help function with this handy utility

By Mark Everingham

In the old Commodore advertisement (you know, the four page epic which managed to link Charles Babbage, an elephant, and a teddy-bear called BJ to buying a Commodore computer for Christmas), special emphasis was laid on the 'HELP' function of the C16 and Plus/4 computers. Commodore claimed it helps you to debug your programs, yet I have owned a Plus/4 for several months now, and can honestly say I have never used the HELP function except for the novelty. The BBC Micro has a command *HELP which lists the syntax of a given command on a sideways ROM, and I decided to implement such a function on the Plus/4. I decided on three features it should have –

1. It should be compatible with the Commodore C16.

2. It should not take any memory from the programmer.

3. It should not interfere with the normal HELP function.

A tall order? Well, I decided that to allow a reasonable amount of help on the C16, the program must use the Disk Drive. That way, I could put it in the cassette buffer so as not to use up any memory; and a CHRGET wedge like the DOS SUPPORT program seemed appropriate to allow for the normal HELP function. The result is a 142 Byte piece of machine code using standard PRG files on a Commodore Disk Drive such as the 1551!

The Programs

Listing 1 is a short and sweet Screen Editor I wrote to produce HELP screens and store them on Disk.

Listing 2 is the Basic Loader program for the HELP command. It Pokes all the code into memory and redefines the function keys.

Listing 1 - The Screen Editor

When you've entered and debugged the program, save it onto a disk using DSAVE 'HELP EDITOR' and RUN the program again. You should be presented with a white screen showing the usual flashing cursor in the top-left corner. At this stage, the editor acts just as if you were editing a document – type text, in normal or reverse, graphics symbols, anything you want, even use the ESC functions to format your screen.

Pressing RETURN from the first level puts you into the command mode. A bar will appear at the bottom of the screen with three options. Press the relevant function key to select each one.

LOAD prompts for a filename and attempts to load this HELP screen from the disk.

SAVE prompts for a filename and saves the current screen to disk under that name.

CONT puts you back into level – the edit mode.

When you've created a help screen, save it onto Disk under a suitable and memorable name, and exit the program. The HELP command now has something to work with. If you DIRECTORY the disk, you will see a file 'filename.H' The '.H' designates a HELP file.

Listing 2 - The Basic Loader

Now that you have some data on disk type in Listing 2, the Basic Loader, and save it on disk using DSAVE 'HELP PROGRAM'. When you have run it, try pressing function key 1. You should see the message 'SYS 933:REM ON'. SYS 933 turns the HELP command on, press function key 2 or type SYS 950 to turn it off.

With the HELP Command on, try typing in HELP (RETURN). This



simply does the normal HELP function – if your program has no errors in it, nothing should happen. Now, try typing HELP 'Filename', where filename is the name you stored your Help screen under. If all is well, your screen should load and the READY prompt will appear at the bottom of the screen. If this does not happen, something is wrong – if you get an error message '?STRING TOO LONG ERROR' this means that you have tried to type in a name more than 14 characters long. If you get the error message '?DIRECT MODE ONLY ERROR' then you have tried to use the new HELP command from within a program. If neither of these, use PRINT DS$ to find the error The Syntax, and errors returned by the HELP command are shown below.

Syntax of the HELP Command

HELP (RETURN) – Normal HELP function
HELP (Filename) – Loads help file called 'Filename' from disk
SYS 933 – Turns HELP Command On
SYS 950 – Turns HELP Command Off

Errors returned by the HELP Command

All DOS Errors – a fault concerning the disk
?STRING TOO LONG – You have typed a filename longer than 14 Characters
?DIRECT MODE ONLY – You have tried to use HELP in a program

Information on the HELP Program

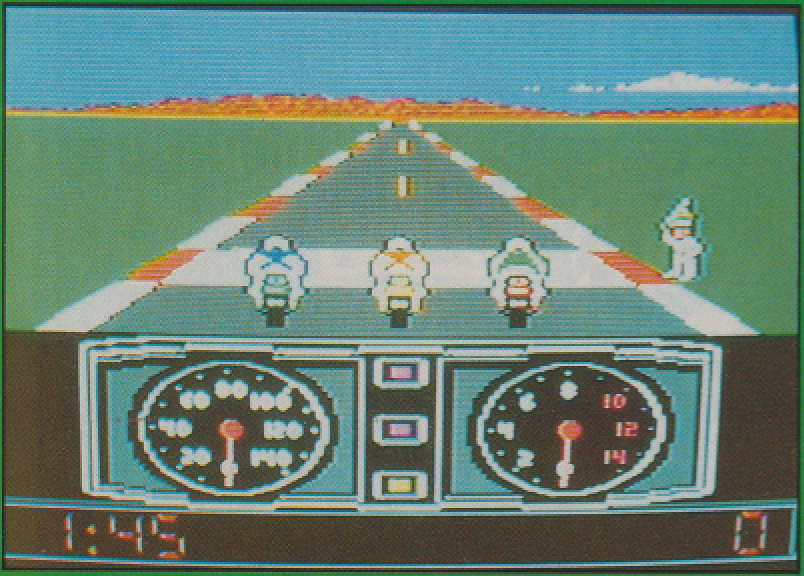
The HELP Command resides by default in the first 142 bytes of the cassette buffer ($332-$3BF). Note that when it is installed in this area of memory, the program will be erased by pressing 'RESET'. I placed it here to avoid clashes between the Plus/4 and the C16. However it may be relocated by changing the address A in line 0 of the BASIC Loader Program.

Practical use of the HELP Command

When creating HELP files, try to give them memorable and unambiguous names. It is also a good idea to make them short, though the '*' and '?' designators may be used in filenames. For instance, if you are going to replace the manual with a HELP disk, divide the files into commands, each showing the syntax and a few examples of the command's use. E.g. Typing HELP 'CIRCLE' might bring up all the different ways of using the command CIRCLE. The thing of utmost importance is to 'use your common sense!' Its a system which is very powerful when used properly, but could end up not being helpful at all if help files are designed without thinking. Anyway, I hope it will be very useful to you!

GAMES UPDATE

Super Cycle



The budget bonanza continues with this classic bike racing game that topped the charts in its full-priced format. Myriad Courses wait to test man and machine in a battle against other riders and the clock, as you have to cross the line before the seconds run out to qualify for the next course.

The early tracks are easy enough, and although the other riders can get in your way and even cause a crash, you'll easily catch up the time. Later on things get tricky, with the addition of slippery road surfaces to send you sliding, treacherous icy conditions, obstacles such as barricades and cones to send you crashing, and a tighter time limit where every second will count.

To add variety to the races, the scrolling landscape changes from day to night, country to city, and mountains to desert, the most impressive being the race near Cape Canaveral.

Arcade action at its best.

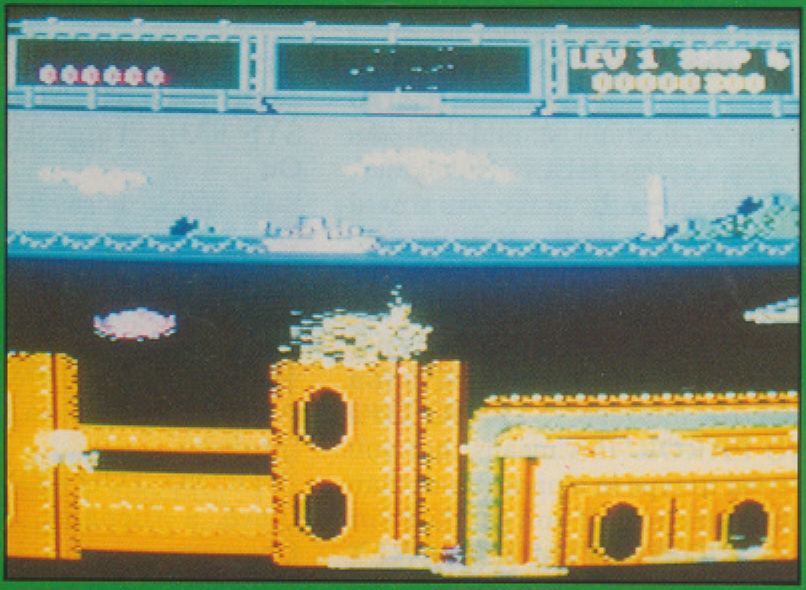
Touchline:
Title: *Super Cycle.* ***Supplier:*** *Kixx (US Gold), Units 2/3, Holford Way, Holford, Birmingham, B6 7AX.* ***Tel:*** *021-356 3388.* ***Price:*** *£2.99.*

The Deep

If you tried to imagine what a game called *The Deep* would be like, you'd probably think of buried wrecks, sharks, octopuses or perhaps submarines. You certainly wouldn't think that you'd spend most of the game on the sea's surface! Unfortunately, this is exactly what you do in this rather disappointing game.

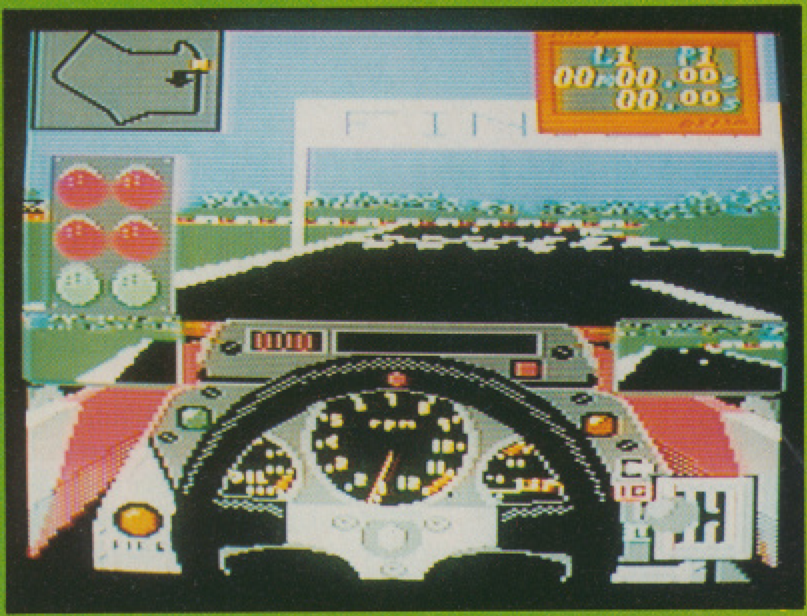
It reminds me of a very old arcade game which I think was called *Submarine Attack*, in which you controlled a ship armed with depth charges that had to survive an assault of submarines that released mines and fired torpedoes at you.

In *The Deep*, your task is to destroy subs, after which small flags are occasionally released and float to the surface, staying there for three seconds. If you manage to avoid the floating mines and reach them in time, a helicopter will fly overhead and drop a package that you must catch. These packages increase your chances in the game by supplying hydrofoil speed to your ship so you can skim across the surface, smart bombs to wipe out all subs, cursor missiles that can be aimed and destroy everything in their path, or submersible pods that can collect the tokens that lie on the sea bed to clear a screen.



There are three transition games in which you must destroy a giant ship, a massive submarine and protect a fleet of escaping hostages by shooting down the missiles that are being fired at them, but then it's back to the main game again.

Touchline:
Title: *The Deep.* ***Supplier:*** *US Gold, Units 2/3 Holford Way, Holford, Birmingham, B6 7AX.* ***Tel:*** *021-356 3388.* ***Price:*** *£9.99.*





GRAND PRIX CIRCUIT

Despite the split between Electronic Arts and Accolade, the string of Accolade imports continues from the EA stable. This latest one attempts to recreate the world of Formula 1 racing, and offers you the chance to drive for either the Ferrari, Williams or Mclaren teams in a world championship against nine other drivers and over eight Grand Prix races.

Selecting the team you will drive for also selects the type of car you will drive – for example, the Ferrari is slower than the others, but is not as likely to spin and so is a good car to start with. When you think you can control the car, you may want to swap to a Williams or the fastest, the McLaren. A more detailed appraisal of each car is presented on-screen in a display that shows the power curves or horse-power and torque, engine size, gear box, chassis and weight, so that people who think these figures are important can use them to decide which car to use in the championship.

Perhaps of more importance is the game level you choose to play at, as this can determine whether it will be a quiet, sporting race without any mechanical problems, or a bitter fight where the best car and driver will win. The problem with the car is keeping it going at a speed fast enough to maintain your race position, but slow enough to stop it from blowing up.

Tyres are also a major headache, particularly in the longer Grand Prix, where cornering too quickly and spinning can cause excess wear. They can be changed in the Pits, but this can cost you valuable seconds if your Pit team isn't quick enough.

The first Grand Prix of the season is at Rio de Janeiro in Brazil, but before you can think about the nine championship points for winning it, you have to drive round the circuit in a qualifying lap. This not only gives you a preview of the track, but the time you take decides your position on the starting grid.

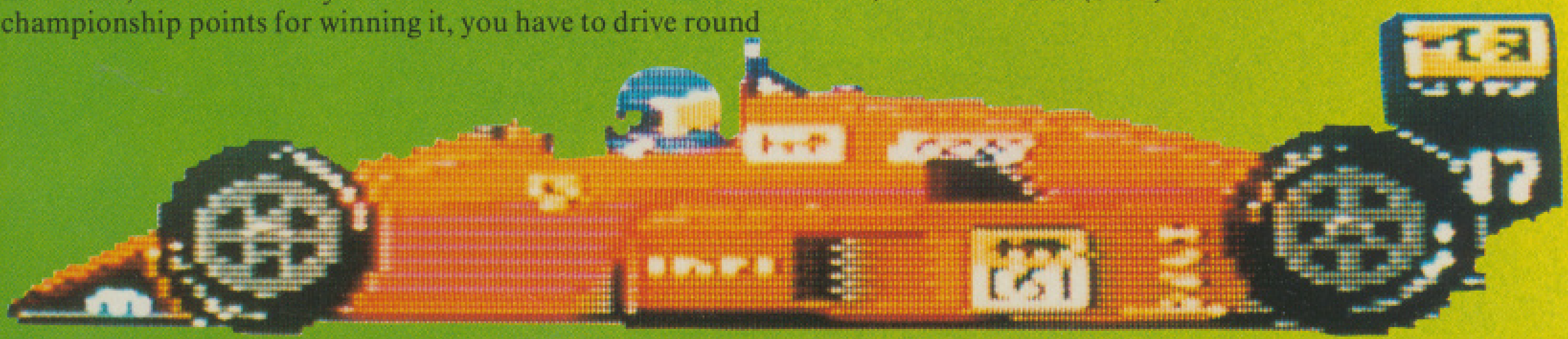
The race screen display shows your view of the track and your controls, which include a tachometer to watch for engine strain, a damage indicator that plots the condition of your car, especially braking and handling, a speedometer, rear view mirrors to check that no-one's in your slipstream, and a map box that includes a line drawing of the course and a flashing dot to plot your position.

Steering can be a little tricky at first, as moving the joystick turns the wheel, and you must remember to turn it back again as it doesn't auto-centre, unlike most driving games. However, once you've mastered the basics you have to contend with the competition, particularly at the higher game levels when any collision will put you out of the race.

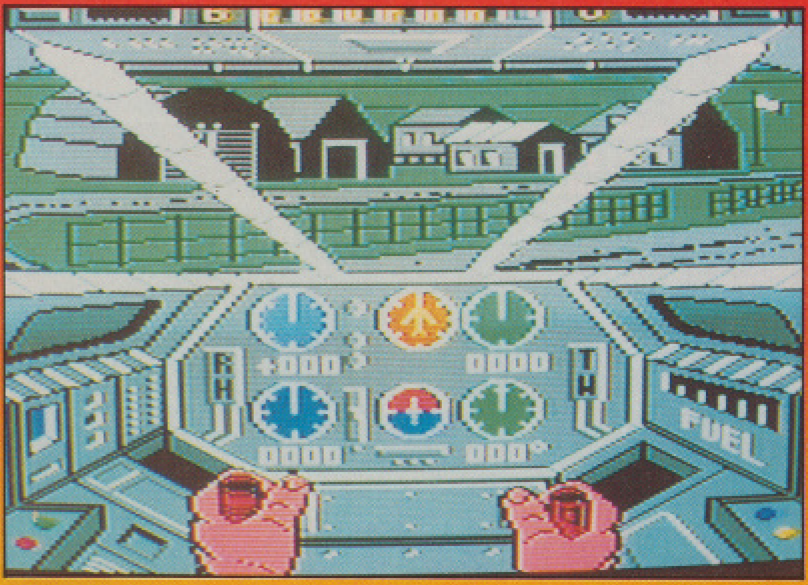
The corners are the best place to overtake, where the driver with the strongest nerve will take the lead – it's all down to when you apply the brakes. If you brake after your opponent, you'll go round the bend in the lead, but if you leave it too late you'll spin off.

While *Grand Prix Circuit* is a good simulation of a formula 1 championship, an eight race season will probably prove too much for all but dedicated formula 1 fans. There is an option to race in a single Grand Prix, but there are better racing games for the casual driver. This one's for those who live on carbon monoxide car fumes.

Touchline:
Title: *Grand Prix Racing.* ***Supplier:*** *Accolade (Electronic Arts), Langley Business Centre, 11-49 Station Rd, Langley, Berks., SL3 8YN.* ***Tel:*** *(0753) 49442.*



Infiltrator



Here's a second chance to fill the ample boots of the one and only Johnny "Jimbo Baby" McGibbets, as he files off in his Whizzbang Enterprises Gizmo chopper to save the world at least three times. The former full-priced game has been re-released via the Kixxs label, and it's excellent value for money.

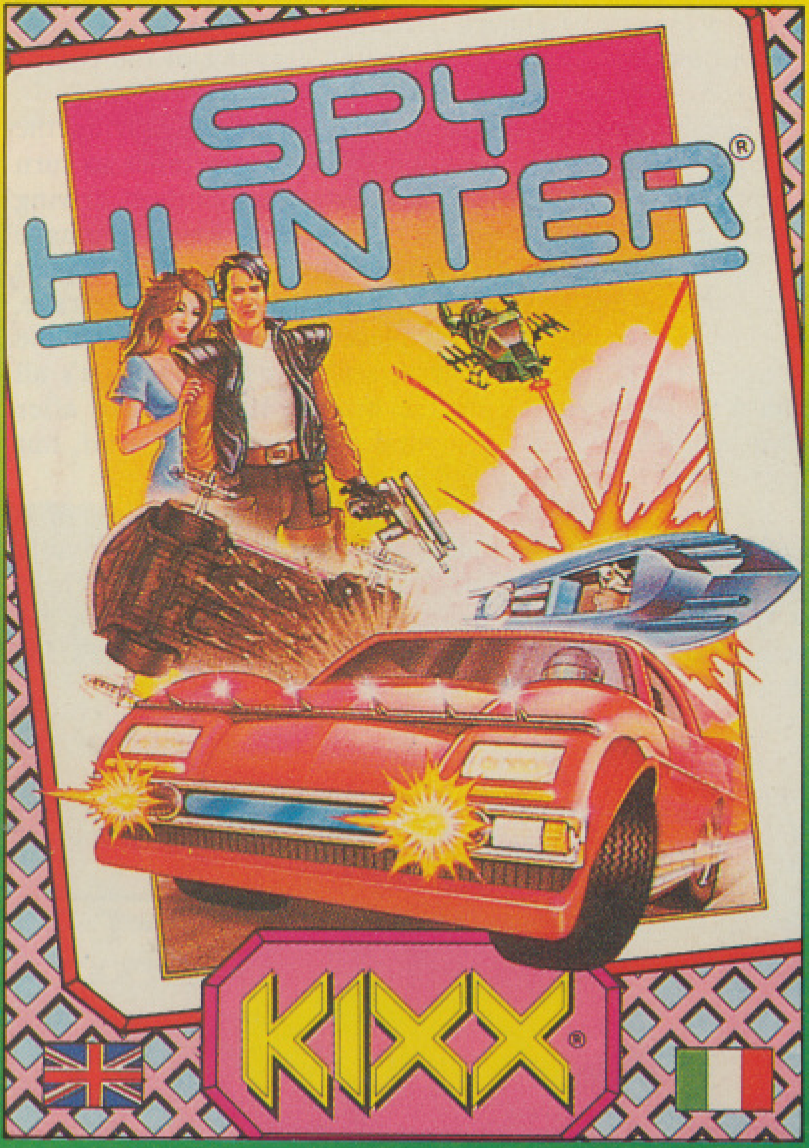
His opponent in all this is the aptly described but unnamed Mad Leader, who is threatening all sorts of despicable things if he isn't stopped, and Jimbo Baby's the man for the job.

Each of the three missions follows a similar pattern – a three stage game in which you must fly through enemy airspace to find the Mad Leader's camp, then (on foot) infiltrate the heavily armed camp and search the buildings for secret plans, weapons and so on.

In fact, *Infiltrator* is three games in one, starting with a combat flight simulator in which you must combine stealth, to confuse the enemy, with fighting skills to shoot them down. In the camp, you are armed with false papers and sleeping gas to get into the camp without raising the alarm. Inside the buildings, things get tougher as you must avoid the guards and search (*Impossible Mission*-style) every object in every room. If the guards get too nosey, showing them your papers may satisfy them, if not, use some sleeping gas and then get out before the come round and raise the alarm.

Infiltrator was a smash hit in the States, but is still underrated over here. At £2.99, it's a bargain.

Touchline:
***Title:** Infiltrator. **Supplier:** Kixx (US Gold), Units 2/3, Holford Way, Holford, Birmingham, B6 7AX. **Tel:** 021 356 3388. **Price:** £2.99.*



This is the re-release of the conversion of the original arcade game that began the craze for car shoot-em-ups that has resulted in a string of games including *Road Blasters* and *LED Storm*. It all started in 1983 with *Spy Hunter*.

Naturally, it looks a little dated with its top-down view of a scrolling road network that you must patrol, but the gameplay is just as addictive, and has certainly stood the test of time.

The action begins as the weapons van pulls up at the side of the road and the Spy car rolls out into your joystick control and begins with a machine gun as its only weapon. Your mission is to stay alive as long as possible (oddly enough) and clear the road of the villains – Road Lords, Barrel Dumpers, Enforcers, Switch Blades, and Mad Bombers (helicopters).

These villains attack you with appropriate weaponry, which you must answer as best you can, while staying on course on a road that turns and splits to confuse you. At some stages the road disappears altogether, but luckily your Spy car then becomes a Spy boat and takes the battle on to the waters.

Destroying enemies not only racks up the points, but also earns you rendezvous with the weapons van and an upgrade for your weapons system, which includes rockets to take out bombing helicopters, and oil slicks and smoke screens to deter enemies that get too close.

Even after six years, *Spy Hunter* is still a great shoot-em-up.

Touchline:
***Supplier:** Kixx (US Gold), Units 2/3, Holford, Holford Way, Birmingham, B6 7AX. **Tel:** 021-356 3388. **Price:** £2.99.*



Denaris is a planet with a problem. For years its scientists have been developing super-advanced machines, and so it was almost inevitable that one day the machines would get so advanced that they wouldn't need the men any more.

By the time the Denarians realised what had happened, it was too late to launch a direct assault. They tried anyway, but it just made the machines more powerful. Their only chance to break the tyranny and escape from their underground prison is you, flying a small fighter.

The fighter is, of course, highly manouverable, and can be improved by collecting items and debris from enemy ships. Before you start thinking that you've heard it all before, and that this is just another *Nemesis* clone, I'm delighted to tell you that it isn't. For one thing, two of you can play together.

The fighter is a DS-H75 Eagle Fighter that begins the game unprepared for the onslaught ahead. You'll face a barrage of meteorites, swarms of mechanised flyers and have to brave the firepower of land-fixed gun emplacements. The other player controls a satellite that's activated by player one collecting a crystal early in the game. From then on, it can either move independently or dock with the fighter (which is what happens in the one player game).

As with *Nemesis*, you can pick up objects to improve your ship, but these are harder to come by. They have specific effects on your ship, and come in the shape of crystals, extra weapons and balls. Crystals are split into two groups, with the Geega crystal activating and improving the firepower of the satellite, and the Zeela stars increasing and decreasing the speed of the alien ships.

There are five types of balls that add a cumulative effect on your ship – they include a red ball to increase shooting power, green to add to the number of missiles, blue to activate a temporary shield, grey to add 1000 bonus points and yellow, which acts as a smart bomb destroying everything currently on the screen.

Extra weapon symbols are less common, and include systems known as the Scatter Shot, Lightening Blitz Shot and Power Shot. You can also collect protection satellites that fly above and below your fighter and deflect incoming missiles. You will need all this firepower and protection if you are to survive the level, as well as the inevitable monster machine that guards the way to the next screen.

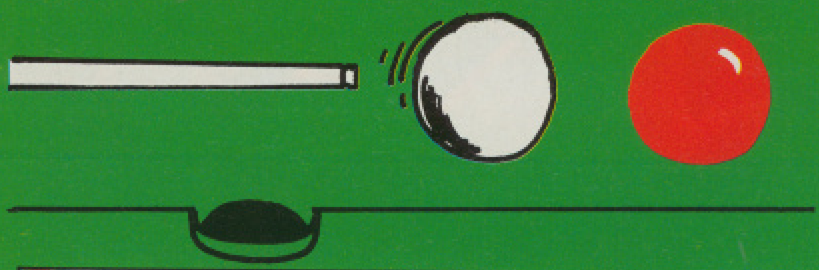
Perhaps the most important thing to learn is control over the fire button, as this is not a game where an auto-firing joystick is the way to get a high score. Instead you must choose the right moment to stab the fire button for a short burst of energy, or press and hold the button to build up the power shot that can take out a medium-sized asteroid or dent a super machine.

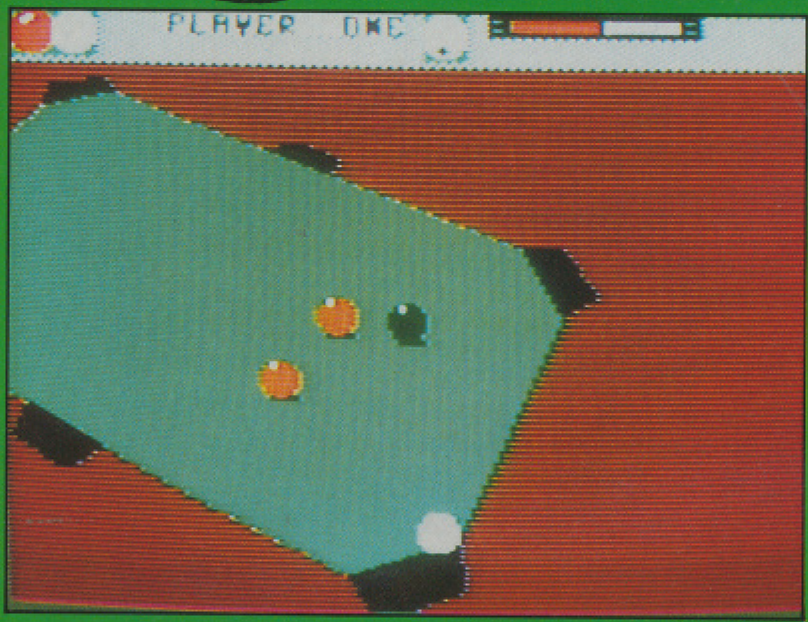
Obviously, *Denaris* is inspired by games like *Nemesis*, but includes enough subtle differences to make it worth a second look.

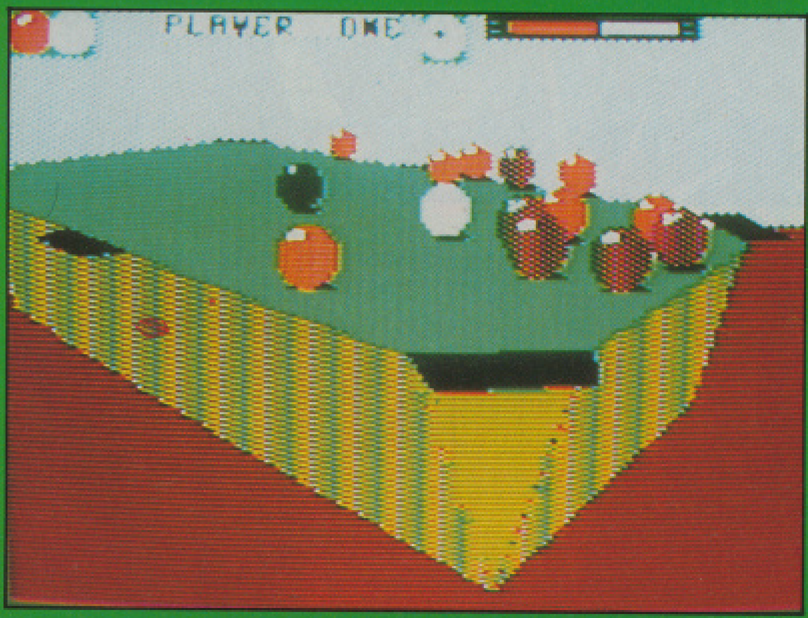
Touchline:

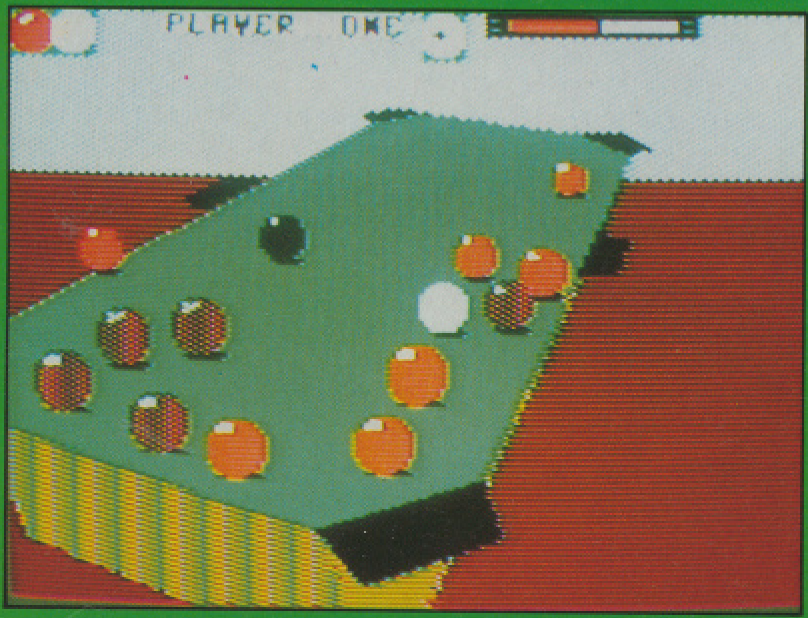
Title: *Denaris.* ***Supplier:*** *Rainbow Arts (US Gold), Units 2/3 Holford Way, Holford, Birmingham, B6 7AX.* ***Tel:*** *021 356 3388.* ***Price:*** *£4.99.*

3D Pool









Pool and snooker games are very popular, but they've always had the problem of being played entirely from a top-down perspective, making it difficult to shoot accurately. Now, thanks to Firebird and the skills of pool champ 'Maltese' Joe Barbara, you can get into the action in one of the first true 3D pool games.

It's quite a remarkable piece of programming – you can walk around the table, pull out for an overhead view and then get down close to aim your shot. There are no aiming lines or cursors, it's up to you to get down close and judge the angles.

You'll need to practise a little on the controls before you can expect to clear the table in a single break. Left and right controls move you around the table, while up and down zoom in and out. Pressing the fire button up and down increases and decreases the power of the shot, moving it left and right controls the horizontal position where the cue will strike the cue ball. This decides the "side" on the ball, and whether it will swerve left or right after it's hit another ball or a cushion.

Top or stun is added by hitting the cue ball above or below its centre, and this, curiously enough, is controlled by the same controls that zoom you in and out. Therefore, to stun the ball back, you have to look down on the table from overhead.

Eventually, you'll sort that one out and be ready to take part in the tournament against seven other players for the right to play Maltese Joe himself. When you realise that Joe is pool's equivalent of Steve Davis, you'll appreciate why you have to qualify first.

This is also the first C64 pool or snooker game which includes an actual tournament where each round is played over more than one frame. For example, you begin your challenge in the quarter final in a best-of-three match, then it's best-of-five in the semis, and seven in the final.

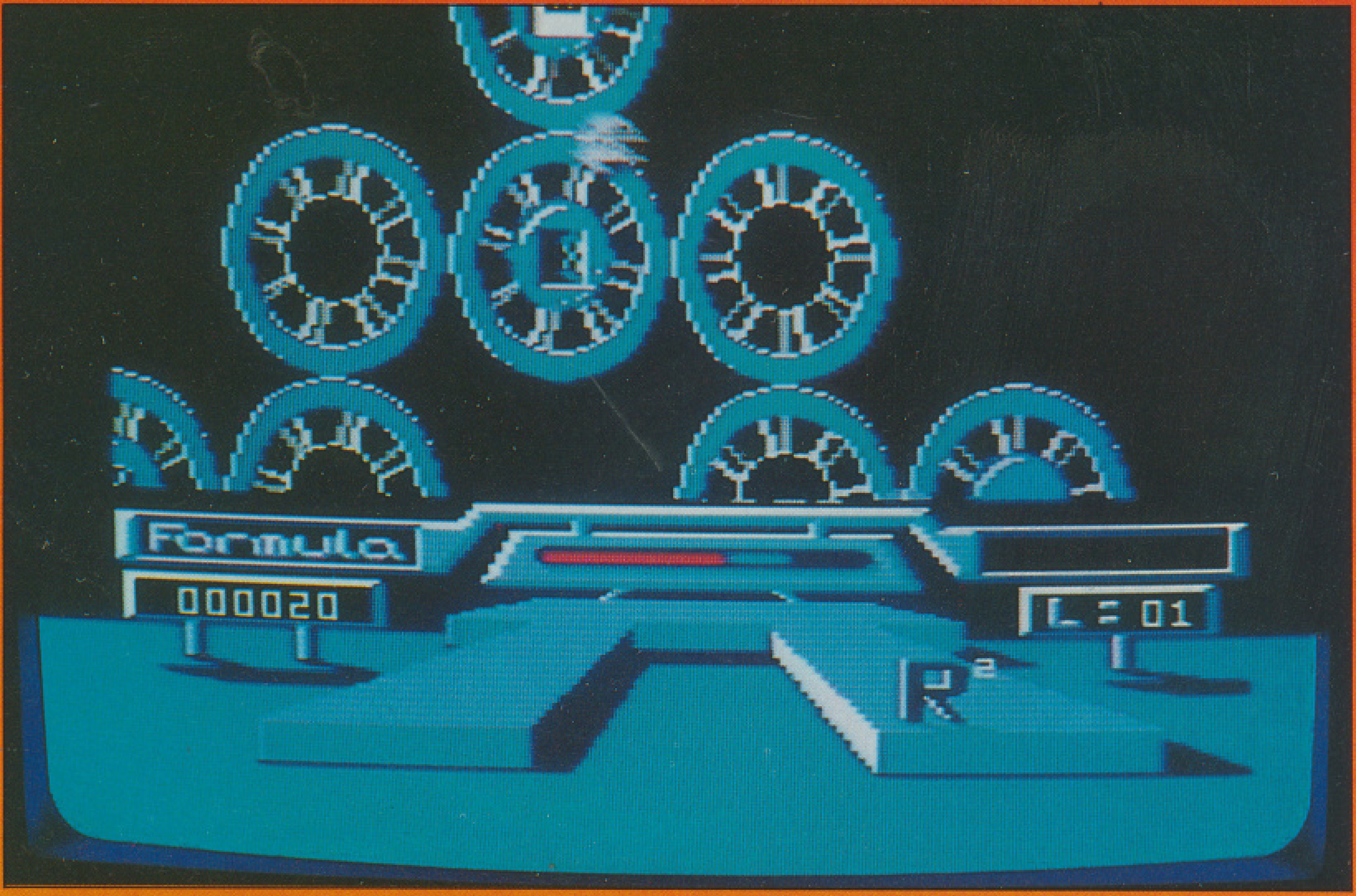
The computer opponents provide a reasonable game, based on the fact that they know what they're doing, whereas you're learning. As you get better, you'll get closer to the challenge match with Maltese Joe. If that's too much for you, or you find it too easy, then you can opt to play against another human, or try one of the 15 trick shots with which you can amaze your friends.

The game plays the popular pub version of pool, in which you must pot all seven of your coloured balls before sinking the black to win the match. However, any foul shots (and the computer opponents occasionally play what I would describe as deliberate misses) are punished by awarding the other player an extra shot and a free ball.

This game is definitely a cut above the usual 2D style of pool game, but it does have some rough edges. Although the balls have shadows and reflections, they slide rather than roll across the table, but that's probably more to do with the technology of the hardware than the programming. The oddest thing is the lack of cushions. Visually, they're three, but the balls seem to drift through them and hang off the edge of the table. However, I doubt there'll be too many complaints about these minor graphical points.

As a first 3D pool game it is exceptional and free from the ridiculous seven cushion shots that plague many 2D versions.

Touchline:
Title: *3D Pool.* ***Supplier:*** *Firebird, 64-76 New Oxford St., London, WC1A 1PS.* ***Tel:*** *01-397 7847.* ***Price:*** *£9.99 (cass), £12.99 (disk).*



Bargain Bucket!

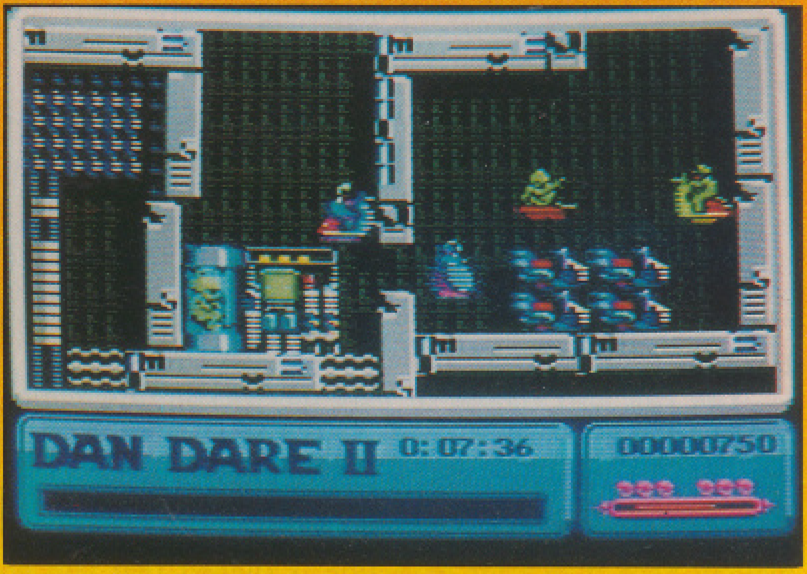
Six new budget games of varying quality are given the once over by ***Gordon Hamlett***

Four titles this month, offering six different games, so everybody should be able to find something to suit both pocket and taste.

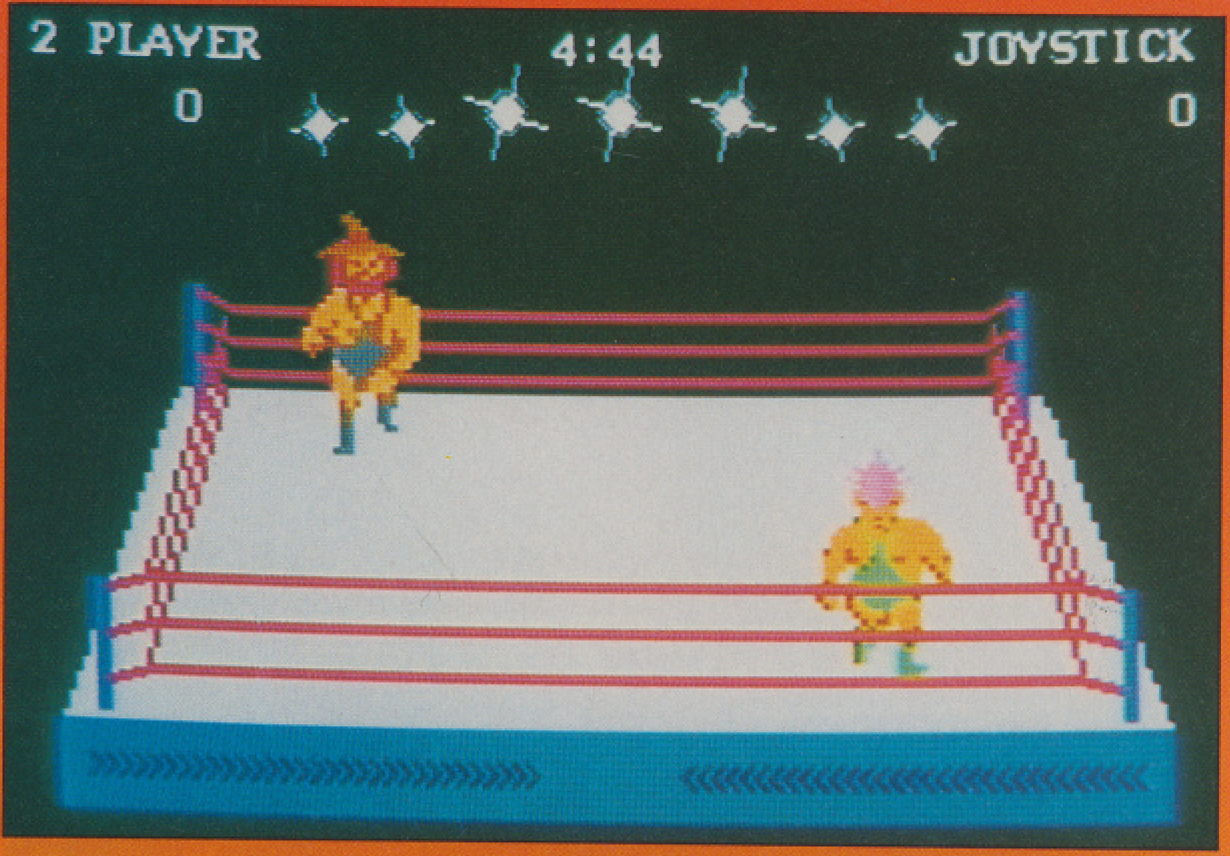
Missing Omega from Bug Byte sees you trying to explore and deactivate an alien object – Omega. The problem is, you only have one hour of real time to accomplish this. There are four different reactors that need to be shut down in this period in order to ensure the safety of life, the universe and everything.

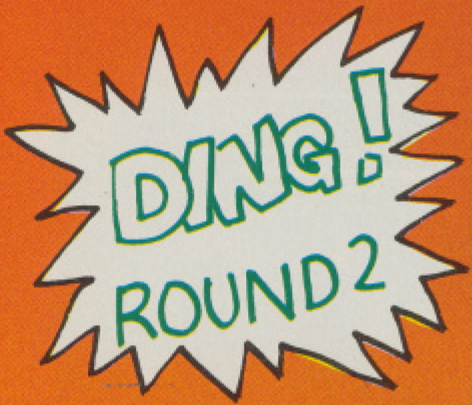
You start off by designing your own robots, each made up of four different components – base, weapon, sensor and power. You must then explore Omega, controlling your robot either manually, on automatic or by programming it.

This is an intriguing game, but it's let down somewhat by an inadequate set of instructions so that even after prolonged playing, I still had little idea of what exactly I was trying to accomplish.



Three games for less than a pound each is the boast of *123* from Silverbird. Pick of these titles is undoubtedly *Rock 'n' Wrestle*, your chance to try out such moves as flying body presses and aeroplane spins on the likes of Vicious Vivian and Gorgeous Greg. Bouncing off the ropes to build up speed, you should attempt to soften your opponent up a bit before attempting to pin him to the canvas.





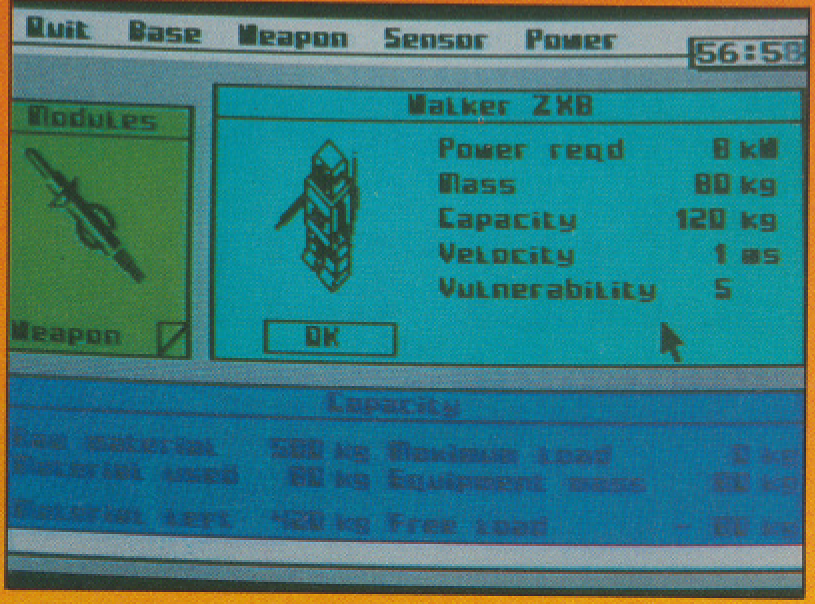
The other two games are somewhat less noteworthy. *BMX Kidz* has you trying to qualify for the next race by finishing in the top three in your current one. It is not sufficient just to complete the course, you have to perform stunts and tricks as well. Collect cans of pop to boost your energy and wheels to repair the damage caused by collision with the other riders.

The final game is *Ninja Master*, which is a truly dreadful martial arts game, but one so bad that it's almost worth having for that fact alone. Dodge arrows, karate chop logs, fend off death stars with your sword and finally shoot down cannisters with your blow-pipe, if you can muster enough interest.

Pi.R.Squared, also from Bug Byte, is a strange game. You must chase round cogs trying to pick up the different elements of assorted mathematical formulae. Pursuing you round the perimeter of the cogs are the chasers. Contact with these decreases your IQ, although you can restore this by collecting books. Other objects may be beneficial or hazardous to your journey. I didn't particularly enjoy this game first time round, and I'm afraid that time has not mellowed my opinion.

The final game this month, and by far the best of the bunch, is *Dan Dare II* on Mastertronic's Ricochet label. Perhaps the most notable aspect of this sequel is that it is actually different from the original – something most unusual in this industry. The evil Mekon has developed a race of Supertreens, and it's up to Dan to penetrate the four levels of the Mekon's ship and sabotage the control boxes.

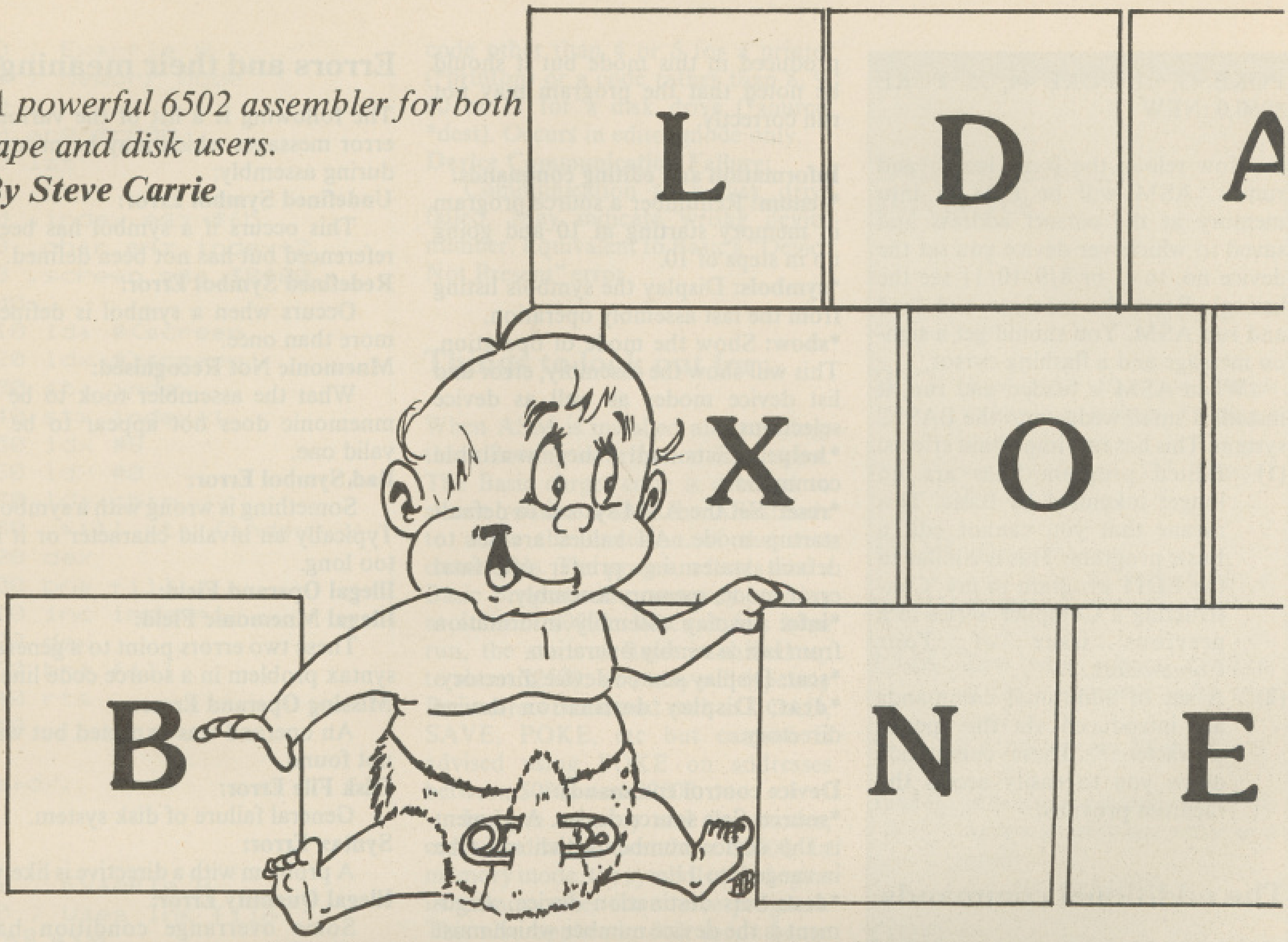
You can also choose to play the Mekon, attempting to release the Supertreen into outer space. In either case, you only have a limited amount of time to accomplish your task before moving onto the next level. This is a good-looking game that plays exceptionally well, and if you don't already have it in your collection, I suggest that you go out and pick up a copy straight away.



Bargain Bucket!

A powerful 6502 assembler for both tape and disk users.

By Steve Carrie



The ASM Assembler

The ASM assembler is a dual mode system whereby 6502 assembly language programs may be compiled from either one or more disk files and/or memory. It provides a set of commands to control its various functions, and also allows output to a Commodore printer.

The system has two modes of operation; disk mode and memory mode. In disk mode, the source code file is read from the source disk drive, and its compiled output sent to another file on the destination disk drive. In memory mode, the source code is read from memory (where it is edited in either mode) and output to memory. Code relocation facilities exist to allow a program to be assembled to run at one address but placed at another.

The ASM system has two error modes, fatal and nonfatal. In fatal mode, an error will cause the assembler to halt, whereas in nonfatal mode, the assembler will process the whole file, listing errors as it goes. Some facilities also exist for control of disk drives and errors reporting from these devices. A printer re-direction facility allows all output which is normally sent to the screen to be sent to the printer at device number 4 or 5. Source and destination drive numbers may be set before assembly commences. Whilst using the editor, any Basic direct mode command may be issued.

Whilst ASM was designed primarily for use with disk drives, the fact that it supports memory assembly allows tape users to make use of it. The memory mode was originally designed to allow short routines to be tested without having to resort to disk usage. Even in memory mode, you may still make use of files from disk as the file-link/file-include facility still works.

Getting it all in

Listed here as a Basic loader program, ASM represents a considerable typing task. You may type in the program directly as it stands, **but don't run it until you have read the next bit!** Before running, (assuming you have saved a copy to tape/disk first) execute these direct BASIC commands first. These will set the memory configuration correctly.

POKE 43, 01 :POKE 44, 35 :POKE 8960,0 :NEW

Now reload the Basic loader and run it. ASM will be POKEd into memory at the correct address and saved to whichever device you set the device no. to (1 or 8/9/10/11 see the listing). Reset the machine and load and run ASM. You should get a sign-on message and a flashing cursor.

When ASM is loaded and run, it installs a small wedge into the BASIC system. This has two important effects:

(1) Edited program lines are no longer tokenised by Basic. This means that you cannot edit a Basic program. This is similar to the EDIT program in my 'Constructing a Compiler' series in a previous issue of *Your Commodore.*

(2) A set of additional commands are introduced via the special character *. These commands allow you to easily access the facilities provided.

The additional commands

The extra commands are as follows:

Assembly:

***assem:** Start the assembler in the mode set by the mode commands *mem and *disk. The operation is as follows. In disk mode, you are asked for a filename whose default extension is .asm. The output file will have extension .exe. In memory mode, the source code is expected from memory and output is to memory. During assembly, output of messages, listings, etc to the list device will follow the mode set by commands *printon and *printoff whilst the treatment of errors will be defined by *fatal and *nonfatal. Assembly may be halted at any time by pressing the RUN/STOP key.

Assembler control commands:

***disk:** Sets assembler disk mode. Source code is expected from disk and output is to a disk file. (see *assem).

***mem:** Sets assembler memory mode. Source code is expected in memory and output is to memory.

***fatal:** If an error is encountered during assembly, the assembler will stop.

***nonfatal:** If an error is encountered during assembly, it is displayed but assembly continues with the next line of source code. Note that output is still produced in this mode but it should be noted that the program may not run correctly.

Information and editing commands:

***renum:** Renumber a source program in memory starting at 10 and going up in steps of 10.

***symbols:** Display the symbols listing from the last assembly operation.

***show:** Show the mode of operation. This will show the assembly, error and list device modes as well as device selections.

***help:** Lists all the available commands.

***reset:** Set the ASM system to default startup mode. All values are set to default states: i.e. printer off, fatal error mode, memory assembly.

***info:** Display assembly information from last assembly operation.

***scat:** Display source device directory.

***dcat:** Display destination device directory.

Device control commands:

***source:** Sets source device. Argument is the device number which must be in range 8 to 11.

***dest:** Sets destination device. Argument is the device number which must be in range 8 to 11.

***scmd:** Send a command to source device. The command must be in quotes. e.g. *scmd "n0:diskl,dl" will format a disk.

***dcmd:** Same as *scmd but for the destination device.

***serr:** Display device error for source device.

***derr:** Display device error for destination device.

***printon:** Enable printer output. Argument is the select code for the list device (4 or 5).

***printoff:** Disable printer output.

Note that certain functions such as listings are controlled from within a source program; e.g. sym, lst. (see directives).

Available Operators.

The following are valid in an expression;

$	Hex value. e.g. sta $C000
'	Ascii value. must be two e.g. lda # 'a'
<	Low byte, e.g. ldx # < symbol
>	High byte e.g. ldy # >symbol
+	Addition e.g. .symbol eqa another+2
-	Subtraction e.g..symbol eqa another-6

Errors and their meanings

The following is a list of the various error messages which may be printed during assembly:

Undefined Symbol Error:
This occurs if a symbol has been referenced but has not been defined.

Redefined Symbol Error:
Occurs when a symbol is defined more than once.

Mnemonic Not Recognised:
What the assembler took to be a mnemonic does not appear to be a valid one.

Bad Symbol Error:
Something is wrong with a symbol. Typically an invalid character or it is too long.

Illegal Operand Field:

Illegal Mnemonic Field:
These two errors point to a general syntax problem in a source code line.

Missing Operand Error:
An operand was expected but was not found.

Disk File Error:
General failure of disk system.

Syntax Error:
A problem with a directive is likely.

Illegal Quantity Error:
Some overrange condition has occurred, typically a 16-bit value in a byte mode instruction.

Illegal Addressing Mode:
An instruction was used in an incorrect way.

Not X or Y Index:
Only X and Y index registers are valid.

Symbol Table Full:
Pretty fatal one this. It indicates that the space set aside for symbols has been exceeded.

Branch Range Error:
Branch instructions are relative and may only operate within a certain range.

Linkfile name length error:
The argument to an lnk directive is too long.

Linkfile name missing:
The argument isn't there at all!

Bad directive in memory mode:
You have used some directive not valid in memory mode.

Bad directive in disk mode:
You have used some directive not valid in disk mode.

Cannot open another linkfile:
Trapping error message when you try to link another file whilst already linked.

No such select code for this device:
You have tried to assign a device

```
10 ; Example A
20 ; Memory Mode
30 ;
40 org $c000
50 lst
60 ;
70 .index eqz $fb
80 .char eqz index+2
90 .screen eqa $0400
100 ;
110 lda #<screen
120 ldx #>screen
130 sta index
140 stx index+1
150 ldx #8
160 ldy #0
170 lda char
180 .fill sta (index),y
190 dey
200 bne fill
210 inc index+1
220 dex
230 bne fill
240 rts
250 ;

ready.
```

```
10 ; Example B
20 ; Uses lnk file
'VAR.ASM'
30 ; use in memeory or
disk mode
40 ;
50 org $c000
60 lst
70 ;
80 lnk "var.asm"
90 ;
100 lda #15
110 ldx #6
120 sta vic+$21
130 stx vic+$20
140 rts
150 ;

ready.
```

Type This In And Save to disk

```
10 ; Module 'VAR.ASM'
20 ; Equate for Example B
30 ;
40 .vic eqa $d000

ready.
```

code other than 4 or 5 for a printer (*printon) or a code rather than 8, 9, 10 or 11 for a disk drive (*source, *dest). Occurs in editor mode only.

Device Communication Failure:
Communication to a disk drive failed. May indicate wrong device number. Equivalent to Basic's "Device Not Present" error.

Things to look out for...

When ASM is installed and running, the following information is relevant. The Basic chrget code is diverted to a new routine within the ASM code to allow the inclusion of the new commands. The program loads like a Basic program into memory starting at address $0801. When it has been run, the start of Basic is shifted up to about $2500. You may still type any Basic direct command such as LOAD, SAVE, POKE, etc but caution is advised using POKE on addresses between $0801 and $2500.

In both memory and disk modes, code is edited above about $2500. In memory mode, the symbol table begins in memory after the source program. This also applies in disk mode hence any program in memory will be preserved. This means that you should type 'new' before commencing a disk mode assembly to maximise symbol space. During assembly, the BASIC ROM is switched out. The space from $C000 to $CFFF is left free. Symbols thus occupy the space from the end of any program in memory up to $BFFF.

ASM should co-exist peacefully with the Basic interpreter. The *reset command may be used to resolve certain situations where the system is not operating correctly. However it has a limited effect, and it may become necessary to powerdown should the system still operate incorrectly.

ASM is source code compatible with my earlier FCL system assembler published in a previous *Your Commodore* and also my PLUS4 assembler. ASM's facilities are in effect, a superset of the FCL assembler's facilities and ASM could therefore replace the FCL assembler if desired.

To help you become familiar with the system, I have included some example source files/listing which may be assembled using ASM. The comment fields at the beginning indicate which mode they should be run in. Good luck!

DMA Assembler Directives

BYT *Byte value directive. Single values or strings in single quotes values.*
e.g. byt 32, 64, $93, 'cbm64', 0

WOR *Word directive*
e.g. wor $C000, $D01A, $3248

EQZ *Zero page equate. Used to assign a zeropage value to a symbol.*
e.g. pointer eqz $FB

EQA *Absolute equate. Used to assign an absolute value to a symbol.*
e.g. vic eqa $D000

ORG *Set code origin. In disk mode – also sets code load address.*
e.g. org $C01A

RES *Reserve memory.*
e.g. res 60 (reserve 60 bytes)

LST *Causes assembler to list during pass 2.*

SYM *Causes assembler to display symbols upon completion of the assembly.*

LNK *Chain to another file. When file has been assembled, the current file resumes assembling.*
e.g. lnk "symbols asm"

REL *Relocation offset. The code origin is set by the org directive. This directive allows you to assemble code to run at one address (org) whilst being sent to another memory area (rel).*
e.g. org $B000
rel $C000

Disk Edit

Delve further into your disks with the help of this article.

By Fergal Moane

Disk editing is what separates a casual disk user from a professional. Once you can edit disks, a whole world of seemingly impossible tasks becomes possible. Files can be locked, unscratched, closed, relocated, and renamed when you have the commands and the know-how.

Firstly, a word of warning: don't edit a disk with important programs on it, unless you know what you're doing. Use unwanted disks for practice, and take backups of valuable disks. A good Disk Editor will make things a lot easier: you have no need for complex commands. There is a good example in the December 1987 edition of *Your Commodore*. This is not essential though - you can make do with the commands and DISPLAY T&S on the demo disk you got with your drive.

Commands

The commands regarding direct disk access are called the Block commands (a Block is another name for a Sector).



Your disk drive manual contains more detailed explanations, but a summary follows.

To use these commands, you'll need to have two files open – one for commands and the other to a buffer for data. The command channel you will probably be familiar with:

OPEN15,8,15

The data channel can be any other number, but 5 or 2 are usually used:

OPEN5,8,5,"#"

After these open commands, PRINT 15 will send commands, and PRINT 5 will send data to the channel.

Note that when 'drive' is mentioned, this means 0 for a single drive. The device is usually 8, but can be changed. See the examples on the disk for more information.

Block-Read

SYNTAX: PRINT #15, "B-R:"; channel; drive; track; sector;

This command transfers the required sector into the data channel (in our case 5). Then use the GET 5 command to read the information into a variable.

It's important to note that Block-Read will only read up as far as the Block-Pointer, which is usually 0. The USER1 command is usually used, as this sets the pointer to 255 automatically, allowing the sector to be read in one operation.

USER1

SYNTAX: PRINT #15, "U1:"; channel; drive; track; sector;

Block-Write

SYNTAX: PRINT #15,"B-W:"; channel;

drive;

track;

sector;

To use this command, fill up the channel with information to write, using PRINT 5, then use the command to write to the required sector. This is the exact opposite of Block-Read, so again USER2 is usually used.

USER2

SYNTAX: PRINT 15, "U2:"; channel; drive; track; sector;

Block-Pointer

SYNTAX: PRINT #15, "B-P:"; channel;

location

By using this command, you can specify where exactly in the sector you want the next read or write to begin. This allows you to read or alter individual bytes in a sector, starting at 'location'. See the Disk Name program for a demo.

Block-Allocate

SYNTAX: PRINT #15,"B-A:"; drive;

track;

sector;

This allocates a bit in the Block Availability Map to show a sector is in use. It is used in connection with random access databases.

Block-Free

SYNTAX: PRINT 15,"B-F:"; drive; track;

sector;

This is the opposite of Block-Allocate, and frees up sectors for use without destroying the actual data on them. If a save is made, the data will probably be overwritten, as the BAM has marked the sector as empty.

Disk Maps

Before you can use these commands, you will need some information on disk structure. The maps will provide this information, and information on file structure.

Editing

There are a number of files provided here for demonstration purposes. The best way to learn is to study these programs with the maps close at hand. They are heavily REMed, but here are some notes explaining what's going on. Even if you don't learn anything, they are useful utilities to have.

Protect File

This program 'locks' (i.e. prevents it from being scratched) the first file on a disk by setting bit 6 of the file type to 1, effectively ORing it with $C0. This prevents accidental erasure, and has a < beside its name in the directory. By adding 32 to the last pointer number, and changing the sector number, any program in the directory may be protected.

Disk Name

This allows you to change the name of the disk without erasing the contents. It makes use of the fact that the disk name is stored at byte 144, track 18, sector 0.

Load Address

This changes the load address of any program to a given address. It searches for the first sector of storage, and bytes 2+3 contain the load address. It is most useful with sprite data.

Unscratch

On scratching a file, the filetype in the directory is merely marked as being deleted. This program searches the disk for a scratched program and restores the filetype, resurrecting the file. You are advised to save the unscratched program to another disk in case of another accident. Note that this will probably not work if something has been saved to the disk since the SCRATCH, as it may have been saved over the old program. Enjoy your disk editing!

BLOCK DISTRIBUTION BY TRACK

Track number	Block range	Total
1 to 17	0 to 20	21
18 to 24	0 to 18	19
25 to 30	0 to 17	18
31 to 35	0 to 16	17

1540/1541 BAM FORMAT

Track 18, Sector 0.

BYTE	CONTENTS	DEFINITION
0,1	18,01	Track and block of first directory block.
2	65	ASCII character A indicating 4040 format.
3	0	Null flag for future DOS use.
4-143		Bit map of available blocks for tracks 1-35.
		*1 = available block 0 = block not available (each bit represents one block)

1540/1541 DIRECTORY HEADER

Track 18, Sector 0.

BYTE	CONTENTS	DEFINITION
144-161		Disk name with shifted spaces.
162-163	160	Disk ID.
164		Shifted space
165-166	50,65	ASCII representation for 2A which is DOS version and format type.
166-167	160	Shifted spaces.
177-255	0	Nulls, not used.

Note: ASCII characters may appear in locations 180 thru 191 on some diskettes.

SEQUENTIAL FORMAT

BYTE	DEFINITION
0-1	Track and block of next sequential data block.
2-256	265 bytes of data with carriage return as record terminators.

PROGRAM FILE FORMAT

BYTE	DEFINITION
0,1	Track and block of next block in program file.
2-256	256 bytes of program info stored in CBM memory format (with key words tokenised). End of file is marked by three zero bytes.

RELATIVE FILE FORMAT

DATA BLOCK

BYTE	DEFINITION
0,1	Track and block of next data block.
2-256	254 bytes of data. Empty records contain FF (all binary ones) in the first byte followed by 00 (binary all zeros) to the end of the record. Partially filled records are padded with nulls (00).

SIDE SECTOR BLOCK

BYTE	DEFINITION
0-1	Track and block of next side sector block.
2	Side sector number (0-5).
3	Record length.
4-5	Track and block of first side sector (number 0).
6-7	Track and block of second side sector (number 1).
8-9	Track and block of third side sector (number 2).
10-11	Track and block of fourth side sector (number 3).
12-13	Track and block of fifth side sector (number 4).
14-15	Track and block of sixth side sector (number 5).
16-256	Track and block pointers to 120 data blocks.

Listings

Help Program

```
PROGRAM:HELP PROG-LISTING1
```

```
10 TRAP 110
20 IF S=0 THEN 210
30 IF C=1 THEN C=0:GOTO 160
40 FOR N=1 TO 3:KEY N,CHR$(N):NEXT
50 SCNCLR:POKE 2021,22:POKE 2022,0
:SYS 55653
60 POKE 2022,23:PRINT"(CLR)(DOWN)"
TAB(10)"(RVS).LOAD (OFF) (RVS) SAV
E (OFF) (RVS) CONT (OFF)";
70 GET K$:IF K$=""THEN 70:ELSE K=A
SC(K$)
80 IF K>3 THEN 70
90 ON K GOTO 140,180,50
100 GOTO 50
110 IF ER=4 THEN RESUME NEXT
120 PRINTCHR$(27)"N?"ERR$(ER)"ERRO
R IN"EL
130 END
140 INPUT"(CLR)LOAD FILE-NAME";F$:
IF LEN(F$)>14 THEN 140
150 C=1:LOAD (F$+".H"),8,1
160 IF DS<20 THEN 50
170 PRINTDS$;:GETKEY G$:GOTO 50
180 INPUT"(CLR)SAVE FILE-NAME";F$:
IF LEN(F$)>14 THEN 180
190 SYS 818 (F$+".H"),8
200 GOTO 160
210 RESTORE:FOR A=818 TO 850:READ
D:POKE A,D:S=S+A:NEXT A:IF S=27522
 THEN 40
220 PRINT"ERROR IN DATA":END
230 DATA 169,0,133,154,32,107,168,
169
240 DATA 0,133,208,169,12,133,209,
169
250 DATA 208,162,231,160,15,32,216
,255
260 DATA 76,251,168,144,3,76,125,1
67,96
```

```
PROGRAM:HELP PROG-LISTING2
```

```
10 RESTORE:FOR L=1 TO 18:S=0
20 FOR N=0 TO 7:READ D$:D=DEC(D$)
30 POKE 810+L*8+N,D:S=S+D:NEXT N
40 READ D$:IF S< >DEC(D$)THEN 80
50 NEXT L
55 KEY 8,"HELP"
60 KEY 2,"SYS 950:REM OFF"+CHR$(13
)
70 KEY 1,"SYS 933:REM ON"+CHR$(13)
:END
80 PRINT"ERROR IN LINE";L*10+90:EN
D
100 DATA A2,22,A5,9A,F0,35,A9,00,3
D1
110 DATA 85,9A,A2,08,A0,03,20,BA,3
46
120 DATA FF,20,64,03,98,A6,22,A4,3
8A
130 DATA 23,20,BD,FF,A9,00,A2,00,3
4A
140 DATA A0,0C,20,D5,FF,20,F8,A8,4
60
150 DATA 90,01,60,A2,16,A0,00,4C,2
95
160 DATA F0,FF,20,48,9C,C9,0F,90,4
5B
170 DATA 05,A2,17,4C,83,86,A8,A9,3
64
180 DATA 2E,91,22,A9,48,C8,91,22,3
4D
190 DATA C8,60,C9,EA,F0,0A,C9,3A,4
D8
200 DATA 90,03,4C,93,04,4C,89,04,2
4F
210 DATA A0,01,B1,3B,F0,11,C9,3A,3
91
220 DATA F0,0D,E6,3B,D0,02,E6,3C,4
12
230 DATA 20,32,03,A9,8F,D0,DF,A9,3
E5
240 DATA EA,D0,DB,A9,B9,8D,AD,03,5
34
250 DATA A2,03,BD,B9,03,9D,84,04,3
43
260 DATA CA,D0,F7,60,A9,BC,D0,ED,6
13
270 DATA 4C,7C,03,C9,3A,B0,00,00,2
7E
```

Line Input

```
PROGRAM:LINE64.BAS
```

```
CC   10 I=51904
18   20 READ A:IF A=256 THEN 6000
     0
57   30 POKE I,A:I=I+1:GOTO 20
A8   51904 DATA 32,121,0,201,46,2
     08,11,32
9A   51912 DATA 115,0,201,133,240
     ,7,201,132
1B   51920 DATA 240,124,76,8,175,
     216,56,32
56   51928 DATA 240,255,142,75,20
     3,140,76,203
81   51936 DATA 169,4,133,252,173
     ,76,203,133
CA   51944 DATA 251,162,40,24,173
     ,75,203,101
6C   51952 DATA 251,133,251,169,0
     ,101,252,133
A9   51960 DATA 252,202,208,240,3
     2,207,255,32
79   51968 DATA 35,203,192,0,208,
     6,32,115
8F   51976 DATA 0,76,121,203,160,
     0,56,177
83   51984 DATA 251,32,57,203,153
     ,61,3,204
CB   51992 DATA 60,3,144,3,76,6,2
     03,200
EF   52000 DATA 76,14,203,172,77,
     203,177,251
8D   52008 DATA 41,127,201,32,208
     ,6,136,240
93   52016 DATA 4,76,38,203,200,1
     40,60,3
9C   52024 DATA 96,41,127,56,201,
     32,144,8
86   52032 DATA 201,64,144,6,24,1
     05,32,96
AB   52040 DATA 105,64,96,0,0,79,
     32,115
D4   52048 DATA 0,32,167,203,32,2
     53,174,166
EE   52056 DATA 20,32,198,255,160
     ,0,32,18
DE   52064 DATA 225,201,13,240,14
     ,200,153,60
4A   52072 DATA 3,165,144,201,64,
     240,52,192
0A   52080 DATA 128,208,235,140,6
     0,3,32,121
AA   52088 DATA 0,32,139,176,133,
     73,132,74
08   52096 DATA 32,163,182,173,60
     ,3,32,117
F6   52104 DATA 180,160,2,185,97,
     0,145,73
C6   52112 DATA 136,16,248,172,60
     ,3,136,185
09   52120 DATA 61,3,145,98,136,1
     6,248,32
A9   52128 DATA 204,255,96,136,76
     ,115,203,32
22   52136 DATA 138,173,76,247,18
     3,0,256
3F   60000 FI$="LINE.EXE"
1F   60010 DE=8
86   60020 POKE194,(51904/256):PO
     KE193,51904-(PEEK(194)*256)
21   60030 POKE175,(52142/256):PO
     KE174,52142-(PEEK(175)*256)
6B   60040 L=LEN(FI$)
38   60050 FORI=1TOL:POKE1023+I,A
     SC(MID$(FI$,I,1)):NEXT
A5   60060 POKE187,0:POKE188,4:PO
     KE183,L:POKE186,DE:SYS62954
```

Analyser

```
PROGRAM:ANALYZER64.PROG1
```

```
72   10 GOSUB900:CLR:REM SET LOAD
      ADDR
E6   15 GOSUB980      :REM SETUP VA
     RIABLES
20   20 GOSUB800      :REM LOAD THE
      ANALYSER
44   30 GOSUB500      :REM RELOCATI
     ON SETUP
12   40 :
5C   50 REM  CHK OP-CODES FOR REL
     OCATION
06   60 :
BF   70 IFCA>EATHENPRINT"RELOCATI
     ON OK":GOTO150
DF   80 I=PEEK(CA):IFOC(I)=0THENP
     RINT"INVALID OP-CODE":END
28   90 IFOC(I)=1THENCA=CA+1:GOTO
     70
FE   100 IFOC(I)=2THENCA=CA+2:GOT
     O70
```

```
B9  110 AD=PEEK(CA+1)+PEEK(CA+2)
    *256:IFAD<A3ORAD>A4THENCA=CA
    +3:GOTO70
5E  115 AD=AD-RF:HB=INT(AD/256):
    LB=AD-HB*256
5A  120 POKECA+1,LB:POKECA+2,HB:
    CA=CA+3:GOTO70
E2  150 PRINT"[DOWN2,RIGHT]1. ST
    ART ANALYSER = SYS";LA
38  160 PRINT"[RIGHT]2. STOP  AN
    ALYSER = SYS";LA+3
71  170 PRINT"[RIGHT]3. DISPLAY
    DATA[SPC3]= SYS";LA+6
2D  180 PRINT
3E  190 END
CF  500 :
6D  510 REM  RELOCATION SETUP
30  520 :
DD  530 IFLA=38912THENCA=1:EA=0:
    RETURN
A8  540 PRINT"RELOCATING ANALYSE
    R"
6D  550 CA=LA:EA=LA+1651:RF=3891
    2-CA
3A  560 PRINT"BUILDING OP-CODE T
    ABLE"
36  565 OC(109)=3:OC(105)=2:OC(9
    7)=2:OC(113)=2:OC(125)=3:OC(
    121)=3:OC(101)=2
87  567 OC(117)=2
3D  570 OC(45)=3:OC(41)=2:OC(33)
    =2:OC(49)=2:OC(61)=3:OC(57)=
    3:OC(37)=2:OC(53)=2
84  580 OC(14)=3:OC(10)=1:OC(30)
    =3:OC(6)=3:OC(22)=2
3A  590 OC(144)=2:OC(176)=2:OC(2
    40)=2:OC(44)=3:OC(36)=2:OC(4
    8)=2:OC(208)=2
0E  600 OC(16)=2:OC(0)=1:OC(80)=
    2:OC(112)=2:OC(24)=1:OC(216)
    =1:OC(88)=1:OC(184)=1
C8  610 OC(205)=3:OC(201)=2:OC(1
    93)=2:OC(209)=2:OC(221)=3:OC
    (217)=3:OC(197)=2
8E  620 OC(213)=2:OC(236)=3:OC(2
    24)=2:OC(228)=2:OC(204)=3:OC
    (192)=2:OC(196)=2
D3  630 OC(206)=3:OC(222)=3:OC(1
    98)=2:OC(214)=2:OC(202)=1:OC
    (136)=1
64  640 OC(77)=3:OC(73)=2:OC(65)
    =2:OC(81)=2:OC(93)=3:OC(89)=
    3:OC(69)=2:OC(85)=2
28  650 OC(238)=3:OC(254)=3:OC(2
    30)=2:OC(246)=2:OC(232)=1:OC
    (200)=1
FA  660 OC(76)=3:OC(108)=3:OC(32
    )=3:OC(173)=3:OC(169)=2:OC(1
    61)=2:OC(177)=2
28  670 OC(189)=3:OC(185)=3:OC(1
    65)=2:OC(181)=2:OC(174)=3:OC
    (162)=2:OC(190)=3
9B  680 OC(166)=2:OC(182)=2:OC(1
    72)=3:OC(160)=2:OC(188)=3:OC
    (164)=2:OC(180)=2
61  690 OC(78)=3:OC(74)=1:OC(94)
    =3:OC(70)=2:OC(86)=2:OC(234)
    =1:OC(13)=3:OC(9)=2
D4  700 OC(1)=2:OC(17)=2:OC(29)=
    3:OC(25)=3:OC(5)=2:OC(21)=2:
    OC(72)=1:OC(8)=1
B7  710 OC(104)=1:OC(40)=1:OC(46
    )=3:OC(42)=1:OC(62)=3:OC(38)
    =2:OC(54)=2
8E  720 OC(110)=3:OC(106)=1:OC(1
    26)=3:OC(102)=2:OC(118)=2:OC
    (64)=1:OC(96)=1
D0  730 OC(237)=3:OC(233)=2:OC(2
    25)=2:OC(241)=2:OC(253)=3:OC
    (249)=3:OC(229)=2
5E  740 OC(245)=2:OC(56)=1:OC(24
    8)=1:OC(120)=1:OC(141)=3:OC(
    129)=2:OC(145)=2
8D  745 OC(157)=3:OC(153)=3:OC(1
    33)=2:OC(149)=2:OC(142)=3:OC
    (134)=2:OC(150)=2
BE  750 OC(140)=3:OC(132)=2:OC(1
    48)=2:OC(170)=1:OC(168)=1:OC
    (186)=1:OC(138)=1
C6  760 OC(154)=1:OC(152)=1
4E  770 PRINT"STARTING RELOCATIO
    N"
81  780 RETURN
19  800 :
A8  810 REM  LOAD THE ANALYSER
0D  820 :
BA  830 LA=PEEK(55)+PEEK(56)*256
01  840 PRINT"[CLR,DOWN] LOADING
     THE ANALYSER AT";LA
94  850 FORI=LATOLA+2047:READQ:P
    OKEI,Q:X=X+Q:NEXT
F7  860 IFX<>244880THENPRINT"LOA
    D FAILED":END
9A  870 PRINT"LOAD OK":RETURN
BD  900 :
65  910 REM 'SETUP ANALYSER LOAD
     ADDR
A1  920 :
11  925 DIMOC(255):I=0:CA=0:AD=0
    :LB=0:HB=0
8D  930 HA=PEEK(55)+PEEK(56)*256
    :HA=HA-2048
CB  940 HB=INT(HA/256):LB=HA-HB*
    256
1C  950 POKE51,LB:POKE52,HB:POKE
    55,LB:POKE56,HB:POKE643,LB:P
    OKE644,HB
4D  960 RETURN
F7  970 :
03  980 DIMOC(255):I=0:CA=0:AD=0
    :LB=0:HB=0: A3=38912:A4=4095
    9:RETURN
5B  1000 DATA 76,9,152,76,27,152
    ,76,39,152,32,162,152,208,9,
    32,166,152,32,196
98  1010 DATA 152,32,212,152,32,
    223,152,96,32,162,152,240,3,
    32,4,153,32,223
6F  1020 DATA 152,96,32,22,153,2
    40,3,32,48,152,96,32,96,157,
    32,177,153,32
92  1030 DATA 231,153,32,5,154,3
    2,106,157,32,83,154,32,230,1
    54,32,255,154
B7  1040 DATA 32,10,155,32,21,15
    5,32,243,157,32,212,154,32,1
    58,155,32,167
8A  1050 DATA 155,32,200,155,32,
    243,157,32,209,155,32,32,158
    ,32,247,155,32
49  1060 DATA 228,155,32,32,158,
    32,3,156,32,12,188,32,15,156
    ,32,221,189,32
9E  1070 DATA 135,180,32,33,171,
    32,29,156,32,121,156,32,60,1
    57,240,197,32
2B  1080 DATA 64,157,32,96,157,3
    2,116,157,32,159,157,32,189,
    157,32,106,157
DA  1090 DATA 32,237,157,96,173,
    116,158,96,206,116,158,162,1
    ,173,8,3,141
E3  1100 DATA 117,158,173,9,3,15
    7,117,158,173,119,158,141,8,
    3,189,119,158
CB  1110 DATA 141,9,3,96,165,160
    ,141,123,158,165,161,141,124
    ,158,165,162
8F  1120 DATA 141,125,158,96,165
    ,57,141,121,158,165,58,141,1
    22,158,96,165
65  1130 DATA 58,201,250,144,30,
    173,116,158,208,14,162,1,173
    ,177,158,188
77  1140 DATA 177,158,32,30,171,
    76,3,153,162,1,173,175,158,1
    88,175,158,32
6D  1150 DATA 30,171,96,238,116,
    158,162,1,173,117,158,141,8,
    3,189,117,158
92  1160 DATA 141,9,3,96,173,166
    ,153,208,21,173,167,153,201,
    160,208,14,162
E4  1170 DATA 1,173,193,158,188,
    193,158,32,30,171,169,0,96,1
    69,255,96,173
9E  1180 DATA 121,158,197,57,208
    ,10,173,122,158,197,58,208,3
    ,108,117,158
46  1190 DATA 173,128,158,56,233
    ,1,141,128,158,176,20,173,12
    9,158,56,233
79  1200 DATA 1,141,129,158,176,
    9,32,129,153,32,48,152,32,14
    7,153,160,5,162
B2  1210 DATA 0,189,121,158,32,1
    65,153,232,136,208,246,165,5
    7,141,121,158
71  1220 DATA 165,58,141,122,158
    ,32,196,152,108,117,158,162,
    4,189,121,158
40  1230 DATA 157,142,158,202,20
    8,247,173,121,158,141,142,15
    8,96,162,4,189
F7  1240 DATA 142,158,157,121,15
    8,202,208,247,173,142,158,14
    1,121,158,96
71  1250 DATA 141,0,160,238,166,
    153,208,3,238,167,153,96,162
    ,0,189,0,0,157
AA  1260 DATA 0,224,189,0,1,157,
    0,225,189,0,2,157,0,226,189,
    0,3,157,0,227
71  1270 DATA 189,0,4,157,0,228,
    189,0,5,157,0,229,189,0,6,15
    7,0,230,189,0
FD  1280 DATA 7,157,0,231,232,20
    8,205,96,162,0,189,0,216,157
    ,0,232,189,0
20  1290 DATA 217,157,0,233,189,
    0,218,157,0,234,189,0,219,15
    7,0,235,232,208
69  1300 DATA 229,96,173,32,208,
    141,0,236,173,33,208,141,1,2
    36,173,134,2
0A  1310 DATA 141,2,236,173,21,2
    08,141,3,236,173,17,208,141,
    4,236,173,22
AA  1320 DATA 208,141,5,236,173,
    24,208,141,6,236,169,254,141
    ,32,208,169,246
2F  1330 DATA 141,33,208,169,14,
    141,134,2,169,0,141,21,208,1
    69,27,141,17
E3  1340 DATA 208,169,200,141,22
    ,208,169,21,141,24,208,96,32
    ,243,157,32,212
9F  1350 DATA 154,32,228,155,32,
    32,158,32,247,155,173,166,15
    3,174,167,153
80  1360 DATA 56,233,5,141,0,158
    ,176,1,202,142,1,158,32,243,
    157,32,228,155
37  1370 DATA 32,32,158,32,3,156
    ,32,12,188,32,15,156,162,1,1
    73,60,159,188
D6  1380 DATA 60,159,170,32,212,
    187,169,0,141,0,158,169,160,
    141,1,158,173
9A  1390 DATA 166,153,56,237,0,1
    58,141,45,159,173,167,153,23
    7,1,158,141,44
08  1400 DATA 159,169,0,141,43,1
    59,32,32,158,32,247,155,32,1
    2,188,162,1,173
37  1410 DATA 118,159,188,118,15
    9,162,0,32,7,187,32,170,177,
    141,54,159,140
E1  1420 DATA 53,159,96,162,4,18
    9,121,158,157,46,159,202,208
    ,247,173,121
E1  1430 DATA 158,141,46,159,96,
    162,0,169,32,157,0,4,157,0,5
    ,157,0,6,232
A9  1440 DATA 208,244,162,232,15
    7,255,6,202,208,250,96,169,0
    ,141,196,158
8B  1450 DATA 169,9,141,197,158,
    96,24,174,196,158,172,197,15
    8,32,240,255
67  1460 DATA 96,162,8,188,34,15
    9,202,189,34,159,142,195,158
    ,32,30,171,238
FB  1470 DATA 196,158,32,10,155,
    174,195,158,202,208,231,169,
    23,141,196,158
```

```
6F   1480 DATA 32,10,155,162,1,17
     3,108,159,188,108,159,32,30,
     171,238,196,158
26   1490 DATA 32,10,155,162,1,17
     3,110,159,188,110,159,32,30,
     171,169,4,141
47   1500 DATA 196,158,32,10,155,
     169,194,32,210,255,169,17,14
     1,197,158,32
9B   1510 DATA 10,155,169,194,32,
     210,255,169,30,141,197,158,3
     2,10,155,169
69   1520 DATA 194,32,210,255,169
     ,9,141,197,158,238,196,158,1
     73,196,158,201
D7   1530 DATA 23,240,6,32,10,155
     ,76,92,155,169,4,141,196,158
     ,169,9,141,197
BE   1540 DATA 158,32,10,155,96,1
     69,10,141,197,158,32,10,155,
     96,173,121,158
BC   1550 DATA 141,45,159,173,122
     ,158,141,44,159,169,0,141,43
     ,159,32,32,158
02   1560 DATA 32,247,155,32,221,
     189,32,135,180,32,33,171,96,
     169,18,141,197
53   1570 DATA 158,32,10,155,96,1
     73,48,159,141,43,159,173,49,
     159,141,44,159
20   1580 DATA 173,50,159,141,45,
     159,96,173,123,158,141,43,15
     9,173,124,158
03   1590 DATA 141,44,159,173,125
     ,158,141,45,159,96,162,1,173
     ,140,158,188
F3   1600 DATA 140,158,32,162,187
     ,96,162,1,173,140,158,188,14
     0,158,32,80,184
EB   1610 DATA 96,162,1,173,51,15
     9,188,51,159,162,0,32,7,187,
     96,169,9,141
3E   1620 DATA 197,158,238,196,15
     8,173,196,158,201,23,176,4,3
     2,10,155,96,169
31   1630 DATA 0,32,228,255,240,2
     51,201,136,240,45,201,135,24
     0,7,201,133,240
FA   1640 DATA 50,76,48,156,173,0
     ,158,56,233,195,176,3,206,1,
     158,141,0,158
CE   1650 DATA 173,1,158,201,160,
     176,10,169,0,141,0,158,169,1
     60,141,1,158
F9   1660 DATA 32,243,157,32,230,
     154,32,255,154,32,10,155,32,
     21,155,96,238
84   1670 DATA 112,159,96,173,1,1
     58,205,167,153,144,10,208,9,
     173,0,158,205
1B   1680 DATA 166,153,176,1,96,3
     2,158,155,32,167,155,32,200,
     155,169,32,32
A5   1690 DATA 210,255,169,48,32,
     210,255,32,29,156,169,194,32
     ,210,255,162
39   1700 DATA 7,169,45,32,210,25
     5,202,208,248,169,194,32,210
     ,255,162,12,169
E1   1710 DATA 45,32,210,255,202,
     208,248,238,196,158,32,10,15
     5,173,196,158
67   1720 DATA 201,23,144,12,32,2
     30,154,32,255,154,32,10,155,
     32,21,155,173
67   1730 DATA 53,159,141,121,158
     ,173,54,159,141,122,158,32,1
     58,155,32,167
BC   1740 DATA 155,32,200,155,162
     ,1,173,60,159,188,60,159,32,
     162,187,32,221
79   1750 DATA 189,32,135,180,32,
     33,171,169,0,32,228,255,240,
     251,201,133,240
EA   1760 DATA 41,201,135,240,33,
     201,136,240,3,76,4,157,169,0
     ,141,0,158,169
1B   1770 DATA 160,141,1,158,32,2
     30,154,32,255,154,32,10,155,
     32,21,155,32
23   1780 DATA 101,156,96,32,70,1
     56,96,169,255,141,112,159,96
     ,173,112,159
28   1790 DATA 96,169,102,141,128
     ,158,169,6,141,129,158,169,0
     ,141,166,153
37   1800 DATA 141,0,158,169,160,
     141,167,153,141,1,158,169,0,
     141,112,159,96
B0   1810 DATA 120,173,1,0,41,253
     ,141,1,0,96,173,1,0,9,2,141,
     1,0,88,96,173
0E   1820 DATA 6,236,141,24,208,1
     73,5,236,141,22,208,173,4,23
     6,141,17,208
F7   1830 DATA 173,3,236,141,21,2
     08,173,2,236,141,134,2,173,1
     ,236,141,33,208
4C   1840 DATA 173,0,236,141,32,2
     08,96,162,0,189,0,232,157,0,
     216,189,0,233
A1   1850 DATA 157,0,217,189,0,23
     4,157,0,218,189,0,235,157,0,
     219,232,208,229
B2   1860 DATA 96,162,0,189,0,224
     ,157,0,0,189,0,226,157,0,2,1
     89,0,227,157
6D   1870 DATA 0,3,189,0,228,157,
     0,4,189,0,229,157,0,5,189,0,
     230,157,0,6,189
76   1880 DATA 0,231,157,0,7,232,
     208,211,96,32,228,255,208,25
     1,96,173,1,0
ED   1890 DATA 41,254,141,1,0,162
     ,0,160,5,189,0,160,157,121,1
     58,232,136,208
07   1900 DATA 246,173,0,158,24,1
     05,5,141,0,158,144,3,238,1,1
     58,173,1,0,9
01   1910 DATA 1,141,1,0,96,173,4
     3,159,141,131,158,173,44,159
     ,141,132,158
28   1920 DATA 173,45,159,141,133
     ,158,169,0,141,134,158,169,1
     52,141,130,158
F6   1930 DATA 32,94,158,14,131,1
     58,176,22,206,130,158,14,132
     ,158,144,3,238
5F   1940 DATA 131,158,14,133,158
     ,144,235,238,132,158,24,144,
     229,78,131,158
12   1950 DATA 96,24,173,131,158,
     109,132,158,208,12,176,10,10
     9,133,158,208
79   1960 DATA 5,176,3,238,133,15
     8,96,0,117,158,51,153,0,0,0,
     0,0,0,160,102
F1   1970 DATA 6,0,0,0,0,0,134,11
     2,0,0,0,130,158,0,0,0,0,0,84
     ,82,65,67,69
65   1980 DATA 32,83,84,65,82,84,
     69,68,0,84,82,65,67,69,32,83
     ,84,79,80,80
F6   1990 DATA 69,68,0,147,158,16
     1,158,78,79,32,84,82,65,67,6
     9,32,68,65,84
E4   2000 DATA 65,0,179,158,0,0,0
     ,176,195,195,195,195,195,195
     ,195,178,195
39   2010 DATA 195,195,195,195,19
     5,195,195,195,195,195,195,17
     4,0,194,32,76
4E   2020 DATA 73,78,69,32,32,194
     ,32,69,76,65,80,83,69,68,32,
     32,32,32,194
F0   2030 DATA 0,194,32,32,78,79,
     32,32,32,194,32,84,73,77,69,
     32,40,83,69,67
49   2040 DATA 41,32,194,0,171,19
     5,195,195,195,195,195,195,21
     9,195,195,195
1E   2050 DATA 195,195,195,195,19
     5,195,195,195,195,179,0,0,11
     ,159,244,158
A8   2060 DATA 221,158,198,158,0,
     0,0,0,0,0,0,0,135,158,0,0,0,
     0,0,0,0,55,159
95   2070 DATA 19,173,195,195,195
     ,195,195,195,195,177,195,195
     ,195,195,195
5C   2080 DATA 195,195,195,195,19
     5,195,195,189,0,32,70,49,61,
     69,78,68,32,70
21   2090 DATA 53,61,85,80,32,70,
     55,61,68,79,87,78,0,63,159,8
     6,159,0,131,32
32   2100 DATA 0,0,0,113,159,0,0,
     117,85,0,1,247,245,138,138,2
     55,255,138,138
D7   2110 DATA 255,255,138,138,25
     1,255,138,138,255,255,138,13
     8,255,255,138
13   2120 DATA 138,255,255,138,13
     8,255,255,138,138,255,255,0,
     0,117,117,0,0
8B   2130 DATA 117,117,0,0,117,11
     7,0,0,117,117,0,0,117,117,0,
     0,117,117,0,0
03   2140 DATA 117,117,0,0,117,11
     7,138,138,255,255,138,138,25
     5,255,138,138
29   2150 DATA 255,255,138,138,25
     5,255,138,138,255,139,202,13
     8,255,254,138
89   2160 DATA 138,255,255,138,13
     8,255,155,0,0,117,117,0,0,11
     7,117,0,0,117
B0   2170 DATA 117,0,130,117,119,
     0,0,117,117,0,0,247,117,0,0,
     117,117,0,3,245
66   2180 DATA 247
```

PROGRAM:PANAL64.LDR

```
68   10 LD$=CHR$(34)+"C64PANAL"+C
     HR$(34)+",1"
72   20 PRINT"[CLR,DOWN3]LOAD";LD
     $;"[DOWN9,LEFT16]RUN[HOME]"
46   30 FORI=631TO635:POKEI,13:NE
     XT
62   40 POKE198,5
AA   50 POKE49152,0
31   60 POKE641,0:POKE642,192:POK
     E43,1:POKE44,192
6C   70 POKE643,255:POKE644,207:P
     OKE55,255:POKE56,207:NEW
```

PROGRAM:PANAL64

```
70   10 GOSUB10000:GOSUB9000:T1=T
     I:GOTO30
87   20 PC=PC+1:PB=PEEK(PC):RETUR
     N
53   30 LA=PEEK(PC)+PEEK(PC+1)*25
     6
14   40 IFLA=0THENPC=PC+2:T2=TI:G
     OTO1000
BB   50 PC=PC+4:NL=NL+1:PB=PEEK(P
     C)
00   55 LN=PEEK(PC-2)+PEEK(PC-1)*
     256:PRINT"[HOME]";LN
7F   60 IFPB=0THENPC=PC+1:GOTO30
55   70 IFPB=34GOTO150
38   90 IFPB<128GOTO200
F7   100 PB=PBAND127:TK(PB)=TK(PB
     )+1:NT=NT+1
F0   110 IFPB=3GOTO300
7B   115 IFPB=15GOTO350
3E   120 GOSUB20:GOTO60
B8   130 :
C0   150 GOSUB20:IFPB=34GOTO200
82   160 IFPB=0GOTO60
91   170 GOTO150
2F   200 IFPB>64ANDPB<91THENGOSUB
     400:GOTO60
B8   210 GOSUB20:GOTO60
E6   220 :
F6   300 GOSUB20:IFPB=0GOTO60
3E   310 IFPB=58THENGOSUB20:GOTO6
     0
FB   320 GOTO300
71   330 :
C4   350 GOSUB20:IFPB=0GOTO60
56   360 GOTO350
```

```
99  400 X=0:VR$="":VR$=CHR$(PB)
78  410 GOSUB20
38  420 IFPB>64ANDPB<91THENVR$=VR$+CHR$(PB):GOTO410
29  425 IFPB>47ANDPB<58THENVR$=VR$+CHR$(PB):GOTO410
05  430 IFPB=36ORPB=37THENVR$=VR$+CHR$(PB):GOSUB20
C9  440 IFVN$(X)=""THENVN$(X)=VR$:VA(X)=VA(X)+1:NV=NV+1:RETURN
67  450 IFVN$(X)=VR$THENVA(X)=VA(X)+1:RETURN
48  460 X=X+1:IFX<50GOTO440
59  470 RETURN
00  1000 PRINT"[CLR] -- PROGRAM STATISTICS --"
0A  1005 PRINT"[DOWN] PROGRAM NAME[SPC4]= ";NM$
C3  1010 PRINT"[DOWN] PROGRAM SIZE[SPC4]=";PC-2049
86  1020 PRINT"[DOWN] NO OF LINES[SPC5]=";NL
92  1030 PRINT"[DOWN] NO OF COMMANDS  =";NT
68  1035 PRINT"[DOWN] NO OF VARIABLES =";NV
00  1040 GOSUB1150
74  1060 X=0:PRINT"[CLR]- COMMANDS -"
5E  1070 FORI=0TO127:IFTK(I)=0ANDI>74THENNEXT:GOTO1120
6D  1075 IFTK(I)=0THENGOSUB3000:NEXT:GOTO1120
3E  1080 X=X+1:IFX<23GOTO1110
4B  1100 GOSUB1150:X=0:PRINT"[CLR]";
55  1110 GOSUB3000:GOSUB3100:NEXT
06  1120 GOSUB1150:GOTO1200
42  1140 :
2C  1150 IFPA=1THENRETURN
DB  1160 PRINT"[DOWN]  USE ANY KEY TO CONTINUE";
82  1170 GETR$:IFR$=""GOTO1170
2E  1180 RETURN
8E  1200 :
27  1210 PRINT"[CLR]- VARIABLES -":X=0:J=0
F4  1220 IFX>49GOTO1300
1E  1225 IFVN$(X)=""GOTO1300
3C  1230 PB=LEN(VN$(X)):IFPB>3THENPB=3
F3  1240 ONPBGOTO1250,1260,1270
9B  1250 PRINTVN$(X);"  =";:GOTO1280
89  1260 PRINTVN$(X);" =";:GOTO1280
59  1270 PRINTVN$(X);"=";
AC  1280 PRINTVA(X)
1B  1290 X=X+1:J=J+1:IFJ<22GOTO1220
7D  1294 J=0:GOSUB1150
B2  1298 PRINT"[CLR]";:GOTO1220
25  1300 PRINT"[DOWN] R=RE-DISPLAY, X=END, P=PRINTER";:CLOSE4:PA=0
47  1310 GETR$:IFR$=""GOTO1310
E8  1320 IFR$="R"GOTO1510
FF  1330 IFR$="X"GOTO2000
5E  1340 IFR$="P"THENCLOSE4:GOTO1500
C1  1350 GOTO1310
7F  1500 OPEN4,4:CMD4:PA=1
F4  1510 X=0:TN=41118:GOTO1000
F7  1990 :
50  2000 PRINT"[CLR,DOWN]TIME TAKEN=";(T2-T1)/60:END
83  2990 :
58  3000 TN$="":PB=PEEK(TN)
9C  3010 IFPB>127THENPB=PBAND127:TN$=TN$+CHR$(PB):TN=TN+1:RETURN
40  3020 TN$=TN$+CHR$(PB):TN=TN+1:PB=PEEK(TN):GOTO3010
DB  3030 :
CD  3100 J=LEN(TN$):J=7-J:PRINTTN$SPC(J)"=";TK(I):RETURN
71  9000 :
C6  9020 REM  LOAD BASIC PROGRAM AT 2049
F7  9130 :
F9  9140 INPUT"LOAD FROM DISK (Y/N)";R$
91  9150 IFR$="N"THENDE=1:GOTO9200
E5  9160 IFR$="Y"THENDE=8:GOTO9200
88  9170 GOTO9140
C5  9180 :
B9  9200 INPUT"PROGRAM TO BE LOADED==>";NM$
BE  9210 FL=LEN(NM$)
44  9220 FORJ=1TOFL:POKE511+J,ASC(MID$(NM$,J,1)):NEXT
08  9230 :
31  9300 POKE780,192:SYS65424
03  9310 POKE780,1:POKE781,DE:POKE782,0
BD  9320 SYS65466
AB  9330 POKE780,FL:POKE781,0:POKE782,2
7C  9340 SYS65469
60  9350 POKE780,0:POKE781,1:POKE782,8
E3  9360 SYS65493
39  9370 POKE780,0:SYS65424
CA  9420 :
74  9430 RETURN
3E  9440 :
98  10000 PB=0:PC=2049:NB=0:NL=0:LA=0:DIMTK(127):NT=0:NV=0:X=0:J=0:TN=41118:PA=0
9E  10015 DIMVA(49):DIMVN$(49):VR$="":TN$="":R$="":NM$=""
41  10020 PRINT"[CLR]":RETURN
```

PROGRAM:SWITCH64

```
D1  1000 :
E3  1010 REM THIS PGM PERFORMS 2 FUNCTIONS
90  1020 REM (1) IT IS THE BASIC LOADER FOR
B3  1030 REM     THE SCREEN SWITCH ML PGM &
D4  1040 REM (2) IT INITIALISES 680-727
03  1050 REM     WITH VALUES TO BE USED IN
8F  1055 REM     ZERO PAGE LOCATIONS 43-66
32  1060 REM     FOR THE SCREEN SWITCH ML
42  1065 REM     PGM.
15  1067 :
6D  1070 POKE51,255:POKE52,159:POKE53,255:POKE54,158:POKE55,255:POKE56,159
BB  1075 GOSUB10000          :REM LOAD ML PGM
A8  1080 SA=PEEK(43)+PEEK(44)*256
DC  1090 EA=PEEK(55)+PEEK(56)*256
44  1100 INPUT"ENTER REGION 0 ENDING ADDR -->";R0
E6  1110 IFR0<SAGOTO2050
04  1120 IFR0>EAORR0=EAGOTO2100
59  1130 AD=R0:GOSUB9000
D7  1140 POKE51,LB:POKE52,HB :REM SETUP R0
A1  1150 POKE53,LB:POKE54,HB :REM MEMORY
A0  1160 POKE55,LB:POKE56,HB :REM LIMITS.
22  1170 FORI=0TO23:POKE680+I,PEEK(43+I):NEXT
9A  1180 :
B9  1190 REM SET REGION 1 POINTERS IN 704-727
8E  1200 :
04  1210 POKER0+1,0          :REM FOR BASIC
A7  1220 R1=R0+2             :REM RGN 1 ADDR
AE  1230 AD=R1:GOSUB9000       :REM START OF
76  1240 POKE704,LB:POKE705,HB:REM BASIC.
DE  1250 AD=R1+2:GOSUB9000     :REM START OF
F2  1260 POKE706,LB:POKE707,HB:REM VARIABLS
9F  1270 AD=R1+9:GOSUB9000
7B  1280 POKE708,LB:POKE709,HB:REM ARRAYS
36  1290 POKE710,LB:POKE711,HB:REM START,END
6A  1300 AD=EA:GOSUB9000
51  1310 POKE712,LB:POKE713,HB:REM STRINGS
F3  1320 POKE714,LB:POKE715,HB:REM UTILITY
FD  1330 POKE716,LB:POKE717,HB:REM MEM LIMIT
2B  1340 FORI=718TO725:POKEI,0:NEXT
61  1350 AD=R1-1:GOSUB9000
FF  1360 POKE726,LB:POKE727,HB:REM DATA STM
65  1370 :
FA  1380 PRINT"REGION 0 &[SSPC]1 INITIALISED"
6C  1390 PRINT"REGION ACTIVATED = 0"
18  1400 SYS40704:END
ED  2000 :
DA  2010 REM  ERROR MESSAGES
D1  2020 :
65  2050 PRINT"REGION 0 ENDING ADDR,";R0;"<[SSPC]REGION 0 STARTING ADDR,";SA
29  2060 PRINT"PLEASE REENTER REGION 0 STARTING ADDR"
19  2070 GOTO1100
12  2080 :
90  2100 PRINT"REGION 0 ENDING ADDR,";R0;"> MEMORY LIMIT,";EA
57  2110 GOTO2060
6A  2120 :
72  9000 HB=INT(AD/256):LB=AD-(HB*256):RETURN
0D  10000 :
5E  10010 REM  DATA STATEMENTS FOR ML PGM
71  10020 :
8E  10100 DATA 120,173,20,3,141,171,159,173,21,3,141,172,159,169,39,141,20,3,169
80  10110 DATA 159,141,21,3,162,0,160,48,189,168,2,157,173,159,232,136,208
F6  10120 DATA 246,88,96,165,197,201,64,240,122,201,3,240,7,201,4,240,3,76
C3  10130 DATA 167,159,174,170,159,240,2,162,24,160,0,141,223,159,185,43,0
6E  10140 DATA 157,173,159,232,200,192,24,144,244,173,223,159,201,3,240,8
F8  10150 DATA 162,0,142,170,159,76,102,159,162,1,142,170,159,162,24,160,0
A9  10160 DATA 189,173,159,153,43,0,232,200,192,24,144,244,165,43,141,129
CB  10170 DATA 2,165,44,141,130,2,165,55,141,131,2,165,56,141,132,2,173,170
FC  10180 DATA 159,9,176,141,228,159,172,134,2,162,0,189,224,159,240,13,41
3D  10190 DATA 191,157,35,4,152,157,35,216,232,76,149,159,108,171,159,0,49
BF  10200 DATA 234,1,8,146,49,174,49,79,69,253,79,253,79,255,79,126,4,2,0
D5  10210 DATA 55,33,4,0,0,8,1,80,50,99,50,99,50,99,255,159,255,159,0,160
95  10220 DATA 249,156,0,0,144,0
```

```
    ,0,0,0,80,51,0,4,210,199,206
    ,189,176,0,117
25  10230 DATA 117,0,0,117,117,0
    ,0,117,117,0,0,117,117,128,0
    ,117,117,0,0,117
74  10240 DATA 117,0,0,255,255
28  10250 FORI= 40704TO 40959
25  10260 READJ:X=X+J:POKEI,J:NE
    XT
BF  10270 IFX= 28726THENRETURN
D6  10280 PRINT"DATA STATEMENTS
    INVALID":END
```

Returner

PROGRAM: RETURNER.BAS

```
8F  0 REM
14  1 REM YOU MUST REMEMBER TO R
    AISE BASIC
AD  2 REM BY THE FOLLOWING POKES
    BEFORE
04  3 REM TYPING IN AND RUNNING
    THIS PROG.
F7  4 REM IN DIRECT MODE TYPE 'P
    OKE44,16:'
AE  5 REM POKE4095,0:NEW'
89  6 REM
FC  10 I=2049
2D  20 READ A:IF A=256 THEN 3000
57  30 POKE I,A:I=I+1:GOTO 20
67  2049 DATA 50,8,10,0,158,50,4
    9,49
0E  2057 DATA 48,58,143,34,20,20
    ,20,20
CE  2065 DATA 20,20,20,20,20,20,
    20,20
68  2073 DATA 147,20,40,67,41,49
    ,57,56
F6  2081 DATA 57,32,80,69,84,69,
    82,32
4D  2089 DATA 87,69,73,71,72,73,
    76,76
4E  2097 DATA 0,0,0,0,0,0,0,0
EF  2105 DATA 0,0,0,0,0,169,7,14
    1
99  2113 DATA 134,2,162,0,189,25
    ,8,32
B8  2121 DATA 210,255,232,224,24
    ,208,245,162
3C  2129 DATA 0,160,0,142,32,208
    ,142,33
B9  2137 DATA 208,134,2,232,208,
    253,200,208
39  2145 DATA 250,230,2,165,2,20
    1,4,208
05  2153 DATA 242,162,0,189,128,
    8,157,0
78  2161 DATA 207,232,224,129,20
    8,245,76,0
C3  2169 DATA 207,234,234,234,23
    4,234,234,169
EB  2177 DATA 55,133,1,32,21,253
    ,32,24
F7  2185 DATA 229,169,0,141,24,3
    ,169,207
49  2193 DATA 141,25,3,169,0,141
    ,32,208
5B  2201 DATA 141,33,208,169,128
    ,141,145,2
06  2209 DATA 169,13,32,210,255,
    32,66,166
64  2217 DATA 32,163,253,32,122,
    166,169,8
01  2225 DATA 160,1,162,8,32,186
    ,255,169
A0  2233 DATA 4,162,112,160,207,
    32,189,255
C5  2241 DATA 169,0,170,168,32,2
    13,255,32
7C  2249 DATA 191,227,32,66,166,
    169,8,141
B4  2257 DATA 2,8,32,51,165,165,
    174,133
43  2265 DATA 45,133,47,133,49,1
    65,175,133
38  2273 DATA 46,133,48,133,50,3
    2,89,166
FF  2281 DATA 76,174,167,234,234
    ,234,234,77
74  2289 DATA 69,78,85,160,160,1
    60,160,160
2E  2297 DATA 160,160,160,160,16
    0,160,160,256
C8  3000 POKE43,1:POKE44,8:POKE4
    5,255:POKE46,8
DE  3010 SAVE"RETURNER $CF00",8,
    1
```

PROGRAM:RETURNER-CHECKER

```
E5  10 POKE53280,2:POKE53281,0
AC  20 PRINT"[CLR,C1]CHECKER PRO
    GRAM"
7D  30 PRINT"[YELLOW,DOWN2]FIRST
     TYPE N & RETURN"
4A  40 PRINT"[DOWN]THEN LOAD MEN
    U, LOAD A FILE[SSPC]& RESET
89  50 PRINT"[DOWN]LOAD MENU, LO
    AD ANOTHER FILE, RESET
38  60 PRINT"[DOWN]REPEAT UNTIL
    ALL FILES LOADED
C1  70 PRINT"[DOWN]LOAD CHECKER
    AND TYPE Y
A7  80 INPUT"[DOWN,C1]HAVE YOU W
    IPED MEMORY BEFORE";A$
39  90 IFA$="N"THEN370
B9  100 POKE53280,0
5A  101 INPUT"TO PRINTER";P$
66  102 IFP$="Y"THENP=1:OPEN1,4
F4  110 POKE56334,PEEK(56334)AND
    254
3A  120 J=129:Z=0:F=0
91  130 PRINT"[CLR]POSITIONS FRE
    E[YELLOW,DOWN]"
B7  140 FORI=820TO1023:J=J-1:IFP
    EEK(I)<>252THENJ=129:I=I+120
    :GOTO170
15  150 IFJ>0THEN170
ED  160 F=1:PRINTI-128,:Z=Z+1:IF
    Z=84THENGOSUB310
32  165 IFP=1THENGOSUB500
28  170 NEXT
8A  180 J=129
76  190 FORI=5000TO32768:J=J-1:I
    FPEEK(I)<>252THENJ=129:I=I+1
    20:GOTO220
FF  200 IFJ>0THEN220
55  210 F=1:PRINTI-128,:J=0:I=I+
    99:Z=Z+1:IFZ=84THENGOSUB310
40  215 IFP=1THENGOSUB500
5E  220 NEXT
98  230 J=129
EE  240 FORI=49152TO53247:J=J-1:
    IFPEEK(I)<>252THENJ=129:I=I+
    120:GOTO270
28  250 IFJ>0THEN270
3C  260 F=1:PRINTI-128,:J=0:I=I+
    99:Z=Z+1:IFZ=84THENGOSUB310
15  265 IFP=1THENGOSUB500
8D  270 NEXT
B8  275 IFF=0THENPRINT"[DOWN2]NO
     SPACE FREE IN MEMORY"
7E  290 POKE56334,PEEK(56334)OR1
04  300 CLOSE1:END
5E  310 Z=0:PRINT:PRINT"PRESS AN
    Y KEY TO CONTINUE OR E TO EN
    D[HOME]";:F=1
B8  320 POKE56334,PEEK(56334)OR1
0E  330 GETA$:IFA$=""THEN330
14  340 IFA$="E"THENCLOSE1:END
29  350 POKE56334,PEEK(56334)AND
    254
1D  360 PRINT"[CLR,C1]POSITIONS
    FREE[DOWN,YELLOW]":RETURN
65  370 PRINT"[CLR,DOWN2,YELLOW]
    WIPING":POKE56334,PEEK(56334
    )AND254
3C  380 POKE53265,PEEK(53265)AND
    239
88  390 FORI=820TO1023:POKEI,252
    :NEXT
A7  400 FORI=5000TO32768:POKEI,2
    52:NEXT
5C  410 FORI=49152TO53247:POKEI,
    252:NEXT
FE  420 POKE53265,PEEK(53265)OR1
    6
3A  430 POKE56334,PEEK(56334)OR1
80  440 PRINT"[CLR,DOWN2]LOAD"CH
    R$(34)"MENU"CHR$(34)",8[HOME
    ]";
43  450 END
A1  500 IFJ>0THEN520
D6  510 PRINT#1,I-128;
84  520 RETURN
```

PROGRAM:RETURNER-EDIT

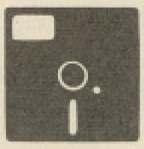
```
48  10 PRINT"[CLR]P[SO]44,10:P[S
    O]2560,0:NEW"
4B  20 PRINT"[DOWN2]LOAD"CHR$(34
    )"EDIT.R"CHR$(34)",8"
1C  30 POKE198,6:POKE631,19:POKE
    632,13:POKE633,13:POKE634,82
    :POKE635,213:POKE636,13
```

PROGRAM:RETURNER-EDIT.R

```
94  10 X=X+1:IFX=1THENLOAD"RETUR
    NER $CF00",8,1
C5  20 POKE53280,0:POKE53281,0
70  30 PRINT"[CLR,C1]","RETURNER
      POSITIONER"
4F  40 PRINT,"[DOWN,LEFT,WHITE](
    C)1989 PETER WEIGHILL[DOWN3]
    "
1C  50 INPUT"[YELLOW]START ADDRE
    SS[SPC3]$334[LEFT6]";SA$
12  53 IFLEFT$(SA$,1)="$"THENGOS
    UB21100:SA=NU:GOTO60
AA  56 SA=VAL(SA$)
55  60 IFSA<820ORSA>53150THEN50
F3  70 INPUT"[DOWN,YELLOW]FILENA
    ME TO LOAD[SPC3]MENU[LEFT6]"
    ;FI$
EA  80 SH=INT(SA/256):SL=SA-(SH*
    256)
33  90 INPUT"[DOWN,YELLOW]FILENA
    ME OF SAVED PROG[SPC3]-[LEFT
    3]";SF$
29  100 S$=""
FA  110 INPUT"[DOWN,YELLOW]ARE Y
    OU SURE ";S$
AF  120 IFS$="N"THEN20
E5  130 LF=LEN(FI$):FI$=LEFT$(FI
    $+"[SSPC17]",16)
CE  140 S1=SH:S2=SL+112:IFS2>255
    THENS2=S2-256:S1=S1+1
4F  150 POKE2161,SH:POKE2169,SH:
    POKE2192,SH:POKE2237,S1
7C  160 POKE2160,SL:POKE2168,SL:
    POKE2187,SL:POKE2235,S2
C9  170 POKE2233,LF
73  180 FORI=2288TO2288+15:POKEI
    ,ASC(MID$(FI$,I-2287,1)):NEX
    T
F4  190 PRINT"[CLR]P[SO]44,8:P[S
    O]43,1:P[SO]46,9:P[SO]45,0:S
    [SA]"CHR$(34)SF$CHR$(34)",8,
    1"
F4  200 POKE198,2:POKE631,19:POK
    E632,13:END
44  21100 NU=0:NU$=RIGHT$(SA$,LE
    N(SA$)-1)
82  21110 IFLEN(NU$)>4THENNU=-1:
    GOTO21180
71  21120 IFLEN(NU$)<4THENNU$="0
    "+NU$:GOTO21120
A8  21130 FORI=1TO4:A$=MID$(NU$,
    I,1):A=VAL(A$)
```

```
E3   21140 IFASC(A$)>64THENA=ASC(
     A$)-55
BB   21150 IF(A$<>"0"ANDA=0)ORA>1
     5GOTO21190
01   21160 N=A*16↑(4-I):NU=NU+N
86   21170 NEXT
80   21180 RETURN
A6   21190 NU=-1:I=4:GOTO21170
```

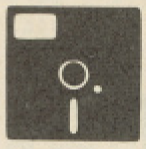
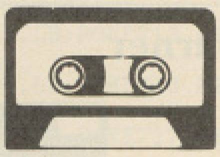
Flow of Ideas



PROGRAM: DIRECTOR

```
BE   100 POKE53280,0:POKE53281,0:
     GOTO440
39   110 GET#8,A$:IFA$=""THENA$=C
     HR$(0)
F6   120 RETURN
55   130 INPUT#15,E,E$,T,S
63   140 IFE>0THENPRINT"[C3]"E;E$
     " "T","S:END
18   150 RETURN
9F   160 GOSUB110
07   170 DN$=DN$+A$
3A   180 RETURN
78   190 GET#8,A$:IFA$=""THEN190
A0   200 IFSTTHEN430
A8   210 IFASC(A$)<128THENF$="DEL
     ":GOTO260
20   220 IFASC(A$)AND56THEN190
BD   230 P$=" ":F=(ASC(A$)AND63):
     F=F+2*(F-1):IFASC(A$)AND64TH
     ENP$="[RVSON]<[RVSOFF]"
E9   240 F$=MID$("SEQPRGUSRREL",F
     ,3)
C9   250 GOSUB110
DA   260 T$=RIGHT$(" "+STR$(ASC(A
     $)),2)
B2   270 GOSUB110
95   280 S$=RIGHT$(" "+STR$(ASC(A
     $)),2)
67   290 N$=""
53   300 FORA=1TO16
28   310 GET#8,A$:N$=N$+A$
C3   320 NEXT
BA   330 FORA=1TO9:GET#8,A$:NEXT
68   340 GOSUB110
17   350 BL=ASC(A$)
54   360 GOSUB110
EF   370 BL=BL+ASC(A$)*256:IFF$<>
     "DEL"THENBT=BT+BL
D5   380 B$=RIGHT$("  "+STR$(BL),
     3)
4C   390 PRINT"[WHITE]";:IFF$="DE
     L"THENPRINT"[C1]";
B2   400 POKE212,128
99   410 PRINTN$;:POKE212,0
A9   420 PRINT"[C8]"P$" [C6]"F$"
      [C3]"B$"[SPC3,C7]"T$"[SPC4]
     "S$
21   430 RETURN
F1   440 PRINT"[CLR,YELLOW,REV N,
     SL]OADING HEADER BLOCK"
8A   450 OPEN15,8,15:PRINT#15,"IO
     ":GOSUB130
CD   460 BT=0:DIMBM$(35)
75   470 OPEN8,8,1,"$,S,R":GOSUB1
     30
E2   480 FORA=1TO2:GET#8,A$:NEXT:
     A=0
14   490 FORA=1TO35
C8   500 GOSUB110
22   510 BM=BM+ASC(A$):BM$(A)=RIG
     HT$(" "+STR$(ASC(A$)),3)
15   520 GET#8,A$:GET#8,A$:GET#8,
     A$
96   530 NEXT
F5   540 DN$="[CLR,REV N,YELLOW,S
     D]ISK: [RVSON]"+CHR$(34)
3E   550 FORA=1TO16:GOSUB160:NEXT
F3   560 DN$=DN$+CHR$(34)
FB   570 FORA=1TO15:GOSUB160:NEXT
8F   580 PRINTDN$:B=0
82   590 PRINT"[DOWN,WHITE,SF]ILE
     NAME[SPC9,C6,ST]YPE [C3,SS]I
     ZE [C7,ST]RACK [SS]ECTOR[DOW
     N]"
82   600 GOSUB190:B=B+1:IFF$<>"DE
     L"THENPT=PT+1
8E   610 IFSTTHEN630
EB   620 IFB<20THEN600
78   630 POKE198,0:PRINTTAB(12)"[
     RED,RVSON] [SP]RESS [SA]NY [
     SK]EY [RVSOFF]";
F9   640 GET K$:IFK$=""THEN640
EA   650 IFST=0THEN580
2D   660 GOSUB130:CLOSE8:CLOSE15
FC   670 PRINT"[CLR,WHITE,SPC11,S
     V]ITAL [SS]TATISTICS"
77   680 PRINT"[SPC11,CT16]"
5F   690 PRINT"[YELLOW,SN]UMBER O
     F PROGRAMS ="PT
FD   700 PRINT"[SB]LOCKS USED ="B
     T
5D   710 BF=BM-VAL(BM$(18))
21   720 PRINT"[SB]LOCKS FREE ="B
     F;
B7   730 A$=""
1B   740 IF664-BT<>BFTHENA$="[C3,
     SB]LOCK COUNT MISMATCH![YELL
     OW]"
8E   750 PRINTA$
B6   760 PRINT"[SD]IRECTORY BLOCK
     S FREE ="BM$(18)
84   770 A$="[GREEN,S-,SPC36,S-]"
07   780 PRINT"[DOWN,GREEN,CA,S*3
     0,CS]"
66   790 PRINT"[GREEN,S-,SF]REE T
     RACK SECTORS:[SPC17,S-]"
63   800 PRINT"[GREEN,S-,CT19,SPC
     17,S-]"
F4   810 PRINT"[GREEN,S-,C8,ST]RA
     CK:  1  2  3  4  5  6  7  8
      9 10[GREEN,S-]"
DD   820 PRINT"[GREEN,S-,C7,ST]OT
     AL:";:FORA=1TO10:PRINT"[C7]"
     ;:IFVAL(BM$(A))=21THENPRINT"
     [C2]";
86   830 PRINTBM$(A);:NEXT:PRINT"
     [GREEN,S-]":PRINTA$
66   840 PRINT"[GREEN,S-,C8,ST]RA
     CK: 11 12 13 14 15 16 17 18
     19 20[GREEN,S-]"
91   850 PRINT"[GREEN,S-,C7,ST]OT
     AL:";:FORA=11TO17:PRINT"[C7]
     ";:IFVAL(BM$(A))=21THENPRINT
     "[C2]";
5E   860 PRINTBM$(A);:NEXT
E0   870 FORA=18TO20:PRINT"[C7]";
     :IFVAL(BM$(A))=19THENPRINT"[
     C2]";
98   880 PRINTBM$(A);:NEXT:PRINT"
     [GREEN,S-]":PRINTA$
04   890 PRINT"[GREEN,S-,C8,ST]RA
     CK: 21 22 23 24 25 26 27 28
     29 30[GREEN,S-]"
DB   900 PRINT"[GREEN,S-,C7,ST]OT
     AL:";:FORA=21TO24:PRINT"[C7]
     ";:IFVAL(BM$(A))=19THENPRINT
     "[C2]";
14   910 PRINTBM$(A);:NEXT
4E   920 FORA=25TO30:PRINT"[C7]";
     :IFVAL(BM$(A))=18THENPRINT"[
     C2]";
AA   930 PRINTBM$(A);:NEXT:PRINT"
     [GREEN,S-]":PRINTA$
6E   940 PRINT"[GREEN,S-,C8,ST]RA
     CK: 31 32 33 34 35[SPC15,GRE
     EN,S-]"
9A   950 PRINT"[GREEN,S-,C7,ST]OT
     AL:";:FORA=31TO35:PRINT"[C7]
     ";:IFVAL(BM$(A))=17THENPRINT
     "[C2]";
50   960 PRINTBM$(A);:NEXT:PRINT"
     [SPC15,GREEN,S-]"
BA   970 PRINT"[C2,S*36,CX]"
```

ASM Assembler

```
85   10 IF PEEK(44)<35 THEN PRINT
      "INSUFFICIENT ROOM. CHANGE
     START OF BASIC!":END
AC   15 AD=2049:PRINT"[CLR,DOWN2]
     LOADING DMA INTO MEMORY"
3C   20 FOR LN=100 TO 4370 STEP 1
     0
40   21 PRINT "[HOME]";LN
5D   25 RT=0
83   30 FOR OS=0 TO 15
2C   35 READ BY:POKE AD+OS,BY
5F   40 RT=RT+BY
AF   45 NEXT
CE   50 READ TT:IF TT<>RT THEN PR
     INT "ERROR IN LINE";LN:END
BC   55 AD=AD+16
BE   60 NEXT
F1   65 PRINT "[DOWN3]SAVING 'DMA
     '"
42   75 POKE 43,1:POKE 44,8:POKE
     45,193:POKE 46,34:SAVE"DMA",
     8
CC   80 CLR
8B   85 PRINT "DONE. NOW RESET SY
     STEM!":END
9D   100 DATA 12,8,10,0,158,32,50
     ,48,54,53,0,0,0,76,198,31, 7
     30
35   110 DATA 76,153,18,0,0,0,0,0
     ,0,0,0,0,0,0,0,0, 247
2B   120 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
F1   130 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
FF   140 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
05   150 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
53   160 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
59   170 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
67   180 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
6D   190 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
3B   200 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
41   210 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
4F   220 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
95   230 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
A3   240 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
A9   250 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
76   260 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
7C   270 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
8A   280 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
D0   290 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
DE   300 DATA 0,0,0,0,0,0,0,0,0,0
     ,0,0,0,0,0,0, 0
```

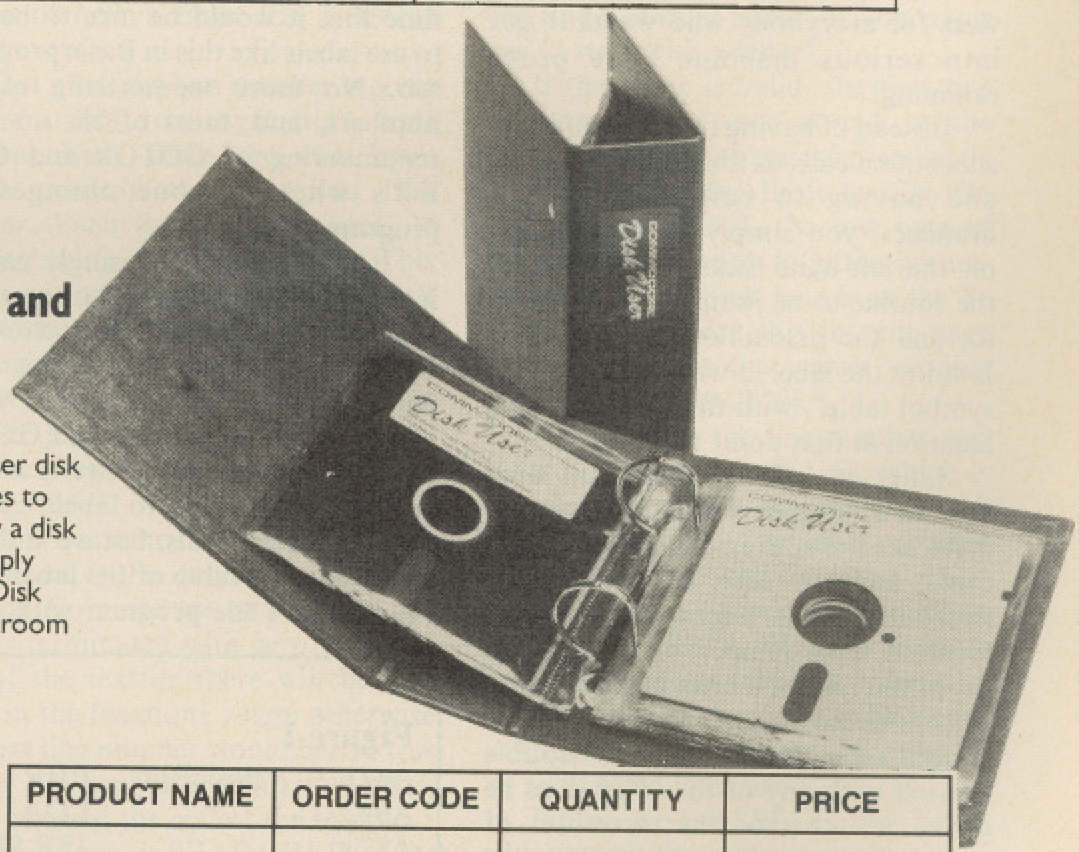
```
E4  310 DATA 0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0, 0
B2  320 DATA 0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0, 0
3E  330 DATA 0,0,0,0,0,0,0,0,0,0,0
    ,0,0,16,39,232,3, 290
E9  340 DATA 100,0,10,0,36,48,56
    ,48,57,49,48,49,49,46,69,88,
    753
5F  350 DATA 69,44,80,44,87,46,6
    5,83,77,64,48,58,195,85,82,8
    2, 1209
B7  360 DATA 69,78,84,32,211,79,
    85,82,67,69,32,196,69,86,73,
    67, 1379
0E  370 DATA 69,61,0,0,13,195,85
    ,82,82,69,78,84,32,196,69,83
    , 1198
A8  380 DATA 84,73,78,65,84,73,7
    9,78,32,196,69,86,73,67,69,6
    1, 1267
B8  390 DATA 0,0,13,0,13,197,78,
    84,69,82,32,198,73,76,69,78,
    1062
B2  400 DATA 65,77,69,32,40,46,6
    5,83,77,41,58,0,13,206,79,32
    , 983
69  410 DATA 70,73,76,69,78,65,7
    7,69,32,71,73,86,69,78,59,32
    , 1077
EE  420 DATA 193,66,79,82,84,73,
    78,71,13,0,13,198,73,76,69,7
    8, 1246
F1  430 DATA 65,77,69,32,84,79,7
    9,32,76,79,78,71,59,32,208,7
    6, 1196
11  440 DATA 69,65,83,69,32,212,
    82,89,32,65,71,65,73,78,13,0
    , 1098
F8  450 DATA 147,13,195,79,77,77
    ,79,68,79,82,69,32,54,52,32,
    196, 1331
FB  460 DATA 85,65,76,32,205,79,
    68,69,32,193,83,83,69,77,66,
    76, 1358
DC  470 DATA 69,82,46,13,210,69,
    83,73,68,69,78,84,32,214,69,
    82, 1341
78  480 DATA 83,73,79,78,32,49,4
    6,49,13,196,73,83,75,32,214,
    69, 1244
A8  490 DATA 82,83,73,79,78,32,5
    0,46,50,13,211,196,195,32,57
    ,46, 1323
5D  500 DATA 56,56,13,13,0,13,13
    ,193,83,83,69,77,66,76,69,82
    , 962
13  510 DATA 32,208,65,83,83,32,
    0,32,66,69,71,73,78,83,46,13
    , 1034
1C  520 DATA 0,13,193,83,83,69,7
    7,66,76,89,32,67,79,77,80,76
    , 1160
9D  530 DATA 69,84,69,46,13,211,
    84,65,82,84,32,65,68,68,82,6
    9, 1191
C8  540 DATA 83,83,32,61,36,0,13
    ,197,78,68,32,65,68,68,82,69
    , 1035
A1  550 DATA 83,83,69,83,32,61,3
    6,0,13,211,89,77,66,79,76,32
    , 1090
41  560 DATA 76,73,83,84,73,78,7
    1,13,0,211,79,85,82,67,69,32
    , 1176
A0  570 DATA 198,73,76,69,32,58,
    32,0,13,196,73,83,75,32,77,7
    9, 1166
6F  580 DATA 68,69,32,83,69,76,6
    9,67,84,69,68,13,0,13,205,69
    , 1054
C9  590 DATA 77,79,82,89,32,77,7
    9,68,69,32,83,69,76,69,67,84
    , 1132
A3  600 DATA 69,68,13,0,13,198,6
    5,84,65,76,32,197,82,82,79,8
    2, 1205
03  610 DATA 32,77,79,68,69,32,8
    3,69,76,69,67,84,69,68,13,0,
    955
F5  620 DATA 13,206,79,78,70,65,
    84,65,76,32,197,82,82,79,82,
    32, 1322
02  630 DATA 77,79,68,69,32,83,6
    9,76,69,67,84,69,68,13,0,212
    , 1135
A1  640 DATA 79,84,65,76,32,197,
    82,82,79,82,83,32,73,78,32,7
    0, 1226
A7  650 DATA 73,76,69,32,61,0,13
    ,13,211,212,207,208,32,75,69
    ,89, 1440
47  660 DATA 32,80,82,69,83,83,6
    9,68,46,13,193,83,83,69,77,6
    6, 1196
A6  670 DATA 76,89,32,65,66,79,8
    2,84,69,68,46,13,0,13,206,79
    , 1067
22  680 DATA 32,83,79,85,82,67,6
    9,32,67,79,68,69,32,70,79,85
    , 1078
CD  690 DATA 78,68,32,73,78,32,7
    7,69,77,79,82,89,13,0,208,82
    , 1137
0F  700 DATA 73,78,84,69,82,32,7
    8,79,84,32,65,67,84,73,86,69
    , 1135
E0  710 DATA 46,13,0,208,82,73,7
    8,84,69,82,32,73,83,32,65,67
    , 1087
04  720 DATA 84,73,86,69,46,13,0
    ,14,147,5,195,79,77,77,79,68
    , 1112
84  730 DATA 79,82,69,32,54,52,3
    2,196,205,193,32,211,89,83,8
    4,69, 1562
2E  740 DATA 77,32,73,78,83,84,6
    5,76,76,69,68,13,212,89,80,6
    9, 1244
E5  750 DATA 32,42,72,69,76,80,3
    2,70,79,82,32,65,32,76,73,83
    , 995
2C  760 DATA 84,32,79,70,32,67,7
    9,77,77,65,78,68,83,13,0,78,
    982
95  770 DATA 69,87,13,196,85,65,
    76,32,205,79,68,69,32,193,83
    ,83, 1435
55  780 DATA 69,77,66,76,69,82,3
    2,40,196,205,193,41,13,210,6
    9,83, 1521
34  790 DATA 73,68,69,78,84,32,2
    14,49,46,49,13,196,73,83,75,
    32, 1234
D6  800 DATA 214,50,46,50,13,196
    ,82,73,86,69,82,83,32,214,52
    ,46, 1388
AF  810 DATA 53,13,195,79,77,77,
    65,78,68,32,214,51,46,48,13,
    211, 1320
69  820 DATA 69,80,84,69,77,66,6
    9,82,32,49,57,56,56,58,211,4
    6, 1161
18  830 DATA 196,46,32,195,65,82
    ,82,73,69,13,0,66,82,75,0,67
    , 1143
3E  840 DATA 76,67,24,67,76,68,2
    16,67,76,73,88,67,76,86,184,
    68, 1379
89  850 DATA 69,88,202,68,69,89,
    136,73,78,88,232,73,78,89,20
    0,78, 1710
E9  860 DATA 79,80,234,80,72,65,
    72,80,72,80,8,80,76,65,104,8
    0, 1327
7F  870 DATA 76,80,40,82,84,73,6
    4,82,84,83,96,83,69,67,56,83
    , 1202
80  880 DATA 69,68,248,83,69,73,
    120,84,65,88,170,84,65,89,16
    8,84, 1627
70  890 DATA 83,88,186,84,88,65,
    138,84,88,83,154,84,89,65,15
    2,66, 1597
B3  900 DATA 67,67,144,66,67,83,
    176,66,69,81,240,66,77,73,48
    ,66, 1456
1D  910 DATA 78,69,208,66,80,76,
    16,66,86,67,80,66,86,83,112,
    65, 1304
C2  920 DATA 68,67,0,65,78,68,11
    ,65,83,76,22,66,73,84,33,67,
    926
0F  930 DATA 77,80,44,67,80,88,5
    5,67,80,89,66,68,69,67,77,69
    , 1143
7D  940 DATA 79,82,88,73,78,67,9
    9,74,77,80,110,74,83,82,121,
    76, 1343
7D  950 DATA 68,65,132,76,68,88,
    143,76,68,89,154,76,83,82,16
    5,79, 1512
56  960 DATA 82,65,176,82,79,76,
    187,82,79,82,198,83,66,67,20
    9,83, 1696
11  970 DATA 84,65,220,83,84,88,
    231,83,84,89,242,66,89,84,0,
    87, 1679
B1  980 DATA 79,82,1,69,81,90,2,
    69,81,65,3,79,82,71,4,82, 94
    0
5C  990 DATA 69,83,5,76,83,84,6,
    83,89,77,7,76,78,75,8,82, 98
    1
CA  1000 DATA 69,76,9,255,105,10
    1,117,255,109,125,121,97,113
    ,255,255,41, 2103
F5  1010 DATA 37,53,255,45,61,57
    ,33,49,255,10,255,6,22,255,1
    4,30, 1437
EF  1020 DATA 255,255,255,255,25
    5,255,36,255,255,44,255,255,
    255,255,255,255, 3650
5F  1030 DATA 201,197,213,255,20
    5,221,217,193,209,255,255,22
    4,228,255,255,236, 3619
42  1040 DATA 255,255,255,255,25
    5,255,192,196,255,255,204,25
    5,255,255,255,255, 3907
2A  1050 DATA 255,255,198,214,25
    5,206,222,255,255,255,255,25
    5,73,69,85,255, 3362
3C  1060 DATA 77,93,89,65,81,255
    ,255,255,230,246,255,238,254
    ,255,255,255, 3158
54  1070 DATA 255,255,255,255,25
    5,255,76,255,255,255,255,108
    ,255,255,255,255, 3754
7E  1080 DATA 255,32,255,255,255
    ,255,255,255,169,165,181,255
    ,173,189,185,161, 3295
FE  1090 DATA 177,255,255,162,16
    6,255,182,174,255,190,255,25
    5,255,255,160,164, 3415
4E  1100 DATA 180,255,172,188,25
    5,255,255,255,74,255,70,86,2
    55,78,94,255, 2982
50  1110 DATA 255,255,255,255,9,
    5,21,255,13,29,25,1,17,255,4
    2,255, 1947
58  1120 DATA 38,54,255,46,62,25
    5,255,255,255,106,255,102,11
    8,255,110,126, 2547
BD  1130 DATA 255,255,255,255,25
    5,233,229,245,255,237,253,24
    9,225,241,255,255, 3952
07  1140 DATA 255,133,149,255,14
    1,157,153,129,145,255,255,25
    5,134,255,150,142, 2963
3D  1150 DATA 255,255,255,255,25
    5,255,255,132,148,255,140,25
    5,255,255,255,255, 3735
40  1160 DATA 255,255,255,65,83,
    83,69,205,68,73,83,203,77,69
    ,205,78, 2126
15  1170 DATA 79,78,70,65,84,65,
    204,70,65,84,65,204,82,69,78
    ,85, 1447
63  1180 DATA 205,83,89,77,66,79
    ,76,211,80,82,73,78,84,79,20
    6,80, 1648
93  1190 DATA 82,73,78,84,79,70,
    198,83,72,79,215,72,69,76,20
    8,82, 1620
50  1200 DATA 69,83,69,212,73,78
    ,70,207,83,69,82,210,68,69,8
    2,210, 1734
```

```
6A  1210 DATA 83,79,85,82,67,197
    ,68,69,83,212,68,67,77,196,8
    3,67, 1583
53  1220 DATA 77,196,83,67,65,21
    2,68,67,65,212,67,76,211,86,
    69,210, 1831
FB  1230 DATA 0,14,8,55,18,58,18
    ,44,18,47,18,119,19,35,18,18
    1, 670
39  1240 DATA 19,216,19,248,19,4
    ,20,213,20,3,32,5,21,10,21,1
    59, 1029
6E  1250 DATA 21,169,21,58,20,63
    ,20,24,21,29,21,228,21,233,2
    1,13, 983
88  1260 DATA 193,83,83,69,77,66
    ,76,89,32,70,65,73,76,83,32,
    79, 1246
5C  1270 DATA 78,32,84,72,73,83,
    32,76,73,78,69,0,213,78,68,6
    9, 1178
5F  1280 DATA 70,73,78,69,68,32,
    211,89,77,66,79,76,32,197,82
    ,82, 1381
0E  1290 DATA 79,82,0,210,69,68,
    69,70,73,78,69,68,32,211,89,
    77, 1344
74  1300 DATA 66,79,76,32,197,82
    ,82,79,82,0,205,78,69,77,79,
    78, 1361
0C  1310 DATA 73,67,32,78,79,84,
    32,82,69,67,79,71,78,73,83,6
    9, 1116
B4  1320 DATA 68,0,194,65,68,32,
    211,89,77,66,79,76,32,69,82,
    82, 1290
CB  1330 DATA 79,82,0,201,76,76,
    69,71,65,76,32,207,80,69,82,
    65, 1330
5C  1340 DATA 78,68,32,198,73,69
    ,76,68,0,201,76,76,69,71,65,
    76, 1296
A3  1350 DATA 32,205,78,69,77,79
    ,78,73,67,32,198,73,69,76,68
    ,0, 1274
82  1360 DATA 205,73,83,83,73,78
    ,71,32,207,80,69,82,65,78,68
    ,32, 1379
75  1370 DATA 197,82,82,79,82,46
    ,0,196,73,83,75,32,198,73,76
    ,69, 1443
16  1380 DATA 32,197,82,82,79,82
    ,46,0,211,89,78,84,65,88,32,
    69, 1316
DF  1390 DATA 82,82,79,82,46,0,2
    01,76,76,69,71,65,76,32,209,
    85, 1331
58  1400 DATA 65,78,84,73,84,89,
    32,197,82,82,79,82,0,201,76,
    76, 1380
AF  1410 DATA 69,71,65,76,32,193
    ,68,68,82,69,83,83,73,78,71,
    32, 1213
42  1420 DATA 77,79,68,69,46,0,2
    06,79,84,32,216,32,79,82,32,
    217, 1398
80  1430 DATA 32,73,78,68,69,88,
    46,0,211,89,77,66,79,76,32,8
    4, 1168
80  1440 DATA 65,66,76,69,32,70,
    85,76,76,46,0,194,82,65,78,6
    7, 1147
F5  1450 DATA 72,32,210,65,78,71
    69,32,69,82,82,79,82,46,0,2
    04, 1273
0A  1460 DATA 73,78,75,70,73,76,
    69,32,78,65,77,69,32,76,69,7
    8, 1090
AD  1470 DATA 71,84,72,32,69,82,
    82,79,82,46,0,204,73,78,75,7
    0, 1199
FF  1480 DATA 73,76,69,32,78,65,
    77,69,32,77,73,83,83,73,78,7
    1, 1109
AC  1490 DATA 46,0,194,65,68,32,
    68,73,82,69,67,84,73,86,69,3
    2, 1108
C8  1500 DATA 70,79,82,32,77,69,
    77,79,82,89,32,77,79,68,69,0
    , 1061
A2  1510 DATA 194,65,68,32,68,73
    ,82,69,67,84,73,86,69,32,70,
    79, 1211
E9  1520 DATA 82,32,68,73,83,75,
    32,77,79,68,69,0,195,65,78,7
    8, 1154
E3  1530 DATA 79,84,32,79,80,69,
    78,32,65,78,79,84,72,69,82,3
    2, 1094
09  1540 DATA 76,73,78,75,70,73,
    76,69,0,206,79,32,83,85,67,7
    2, 1214
B6  1550 DATA 32,83,69,76,69,67,
    84,32,67,79,68,69,32,70,79,8
    2, 1058
6D  1560 DATA 32,84,72,73,83,32,
    68,69,86,73,67,69,46,0,196,6
    9, 1119
7E  1570 DATA 86,73,67,69,32,67,
    79,77,77,85,78,73,67,65,84,7
    3, 1152
FB  1580 DATA 79,78,32,70,65,73,
    76,85,82,69,46,0,64,15,93,15
    , 942
0E  1590 DATA 116,15,139,15,163,
    15,180,15,202,15,225,15,248,
    15,9,16, 1403
F1  1600 DATA 23,16,46,16,71,16,
    89,16,108,16,128,16,156,16,1
    79,16, 928
F6  1610 DATA 209,16,237,16,10,1
    7,47,17,173,112,9,240,82,138
    ,10,72, 1405
2D  1620 DATA 173,27,8,201,1,208
    ,3,76,7,18,238,96,9,208,3,23
    8, 1514
C8  1630 DATA 97,9,32,227,22,104
    ,170,142,23,8,189,78,17,168,
    189,77, 1552
13  1640 DATA 17,170,32,50,22,16
    9,13,32,177,33,173,95,9,240,
    3,76, 1311
F6  1650 DATA 7,18,173,23,8,240,
    5,169,0,76,151,17,32,14,18,3
    2, 983
A4  1660 DATA 215,23,169,0,141,1
    12,9,32,232,17,174,20,8,154,
    96,138, 1540
51  1670 DATA 10,170,189,78,17,1
    68,189,77,17,170,32,50,22,16
    9,13,32, 1403
E2  1680 DATA 177,33,162,128,108
    ,0,3,173,95,9,208,1,96,162,9
    6,160, 1611
FD  1690 DATA 11,32,50,22,173,96
    ,9,174,97,9,133,20,134,21,32
    ,45, 1058
5E  1700 DATA 34,169,13,76,177,3
    3,174,21,8,154,76,189,29,173
    ,93,9, 1428
D5  1710 DATA 240,10,169,2,32,19
    5,255,169,3,32,195,255,169,4
    ,32,195, 1957
0A  1720 DATA 255,96,32,208,23,3
    2,133,32,76,215,23,169,1,44,
    169,0, 1508
FC  1730 DATA 141,95,9,76,88,18,
    169,1,44,169,0,141,93,9,141,
    94, 1288
2C  1740 DATA 9,76,69,18,173,93,
    9,240,7,162,249,160,10,76,50
    ,22, 1423
35  1750 DATA 162,14,160,11,76,5
    0,22,173,95,9,240,7,162,65,1
    60,11, 1417
AC  1760 DATA 76,50,22,162,37,16
    0,11,76,50,22,173,113,9,32,1
    39,18, 1150
4D  1770 DATA 140,179,9,142,180,
    9,173,114,9,32,139,18,140,20
    9,9,142, 1644
00  1780 DATA 210,9,162,157,160,
    9,32,50,22,96,56,233,8,10,17
    0,189, 1573
43  1790 DATA 134,9,168,189,135,
    9,170,96,166,45,164,46,134,4
    3,132,44, 1684
08  1800 DATA 160,0,152,145,43,2
    30,43,208,2,230,44,145,43,20
    0,145,43, 1833
55  1810 DATA 200,24,101,43,133,
    45,165,44,105,0,133,46,32,21
    3,20,32, 1336
AA  1820 DATA 99,166,162,232,160
    ,11,32,50,22,169,8,141,113,9
    ,141,114, 1629
A4  1830 DATA 9,162,0,189,48,12,
    157,119,2,232,224,4,208,245,
    134,198, 1943
0E  1840 DATA 108,2,3,169,166,16
    2,20,141,2,3,142,3,3,169,243
    ,162, 1498
C6  1850 DATA 166,141,6,3,142,7,
    3,96,169,76,162,6,160,19,133
    ,132, 1421
C6  1860 DATA 134,133,132,134,96
    ,201,42,240,7,56,233,48,56,2
    33,208,96, 2049
56  1870 DATA 141,98,9,142,99,9,
    140,100,9,162,0,142,101,9,16
    0,1, 1322
38  1880 DATA 189,164,14,240,69,
    56,241,122,240,19,233,128,24
    0,19,189,164, 2327
D0  1890 DATA 14,48,3,232,208,24
    8,232,238,101,9,76,31,19,232
    ,200,208, 2099
42  1900 DATA 223,200,24,152,101
    ,122,133,122,165,123,105,0,1
    33,123,173,101, 2000
86  1910 DATA 9,10,170,189,18,15
    ,141,100,19,189,19,15,141,10
    1,19,32, 1187
8E  1920 DATA 121,0,32,0,0,162,1
    28,108,0,3,173,98,9,174,99,9
    , .1116
87  1930 DATA 172,100,9,76,10,19
    ,169,10,162,0,133,3,134,4,16
    5,43, 1209
55  1940 DATA 166,44,133,251,134
    ,252,160,1,177,251,208,1,96,
    200,165,3, 2242
32  1950 DATA 145,251,200,165,4,
    145,251,160,0,177,251,72,200
    ,177,251,133, 2582
B1  1960 DATA 252,104,133,251,16
    9,10,24,101,3,133,3,165,4,10
    5,0,133, 1590
CD  1970 DATA 4,76,135,19,32,211
    ,21,169,7,174,116,9,168,32,1
    86,255, 1614
2A  1980 DATA 169,0,32,189,255,3
    2,192,255,176,13,32,183,255,
    208,8,169, 2168
84  1990 DATA 7,141,105,9,76,229
    ,19,169,7,32,195,255,169,0,1
    41,105, 1659
F2  2000 DATA 9,76,229,19,173,10
    5,9,208,7,162,191,160,11,76,
    50,22, 1507
2C  2010 DATA 162,212,160,11,76,
    50,22,32,229,19,32,69,18,32,
    88,18, 1230
CF  2020 DATA 76,107,18,162,0,18
    9,164,14,208,5,169,13,76,177
    ,33,48, 1459
84  2030 DATA 6,32,177,33,232,20
    8,238,41,127,32,177,33,169,1
    3,32,177, 1727
1D  2040 DATA 33,232,208,225,96,
    169,15,168,32,186,255,169,0,
    32,189,255, 2264
99  2050 DATA 32,192,255,144,3,7
    6,109,21,96,174,114,9,208,3,
    174,113, 1723
0F  2060 DATA 9,32,38,20,32,121,
    0,201,34,208,78,160,0,200,17
    7,122, 1432
58  2070 DATA 240,8,201,13,240,4
    ,201,34,208,243,169,0,145,12
    2,165,122, 2115
76  2080 DATA 24,105,1,72,165,12
    3,105,0,72,24,152,101,122,13
    3,122,165, 1486
35  2090 DATA 123,105,0,133,123,
    162,15,32,201,255,104,168,10
    4,170,173,105, 1973
```

```
4F  2310 DATA 121,17,32,179,21,1
    65,3,201,4,144,242,201,6,176
    ,238,141, 1891
6D  2320 DATA 116,9,96,169,147,7
    6,210,255,162,52,160,12,76,5
    0,22,169, 1781
5C  2330 DATA 156,162,12,133,20,
    134,21,162,0,160,2,185,235,8
    ,209,20, 1619
5F  2340 DATA 208,9,136,16,246,1
    60,3,177,20,24,96,232,224,66
    ,176,250, 2043
D0  2350 DATA 165,20,105,4,133,2
    0,165,21,105,0,133,21,24,144
    ,218,173, 1451
DE  2360 DATA 234,8,201,3,240,5,
    162,3,76,121,17,32,240,21,17
    6,246, 1785
B6  2370 DATA 96,134,20,132,21,1
    60,0,177,20,240,6,32,177,33,
    200,208, 1656
B6  2380 DATA 246,96,201,65,144,
    6,201,91,176,2,56,96,24,96,2
    01,48, 1749
4C  2390 DATA 144,250,201,58,176
    ,246,144,242,32,67,22,176,23
    7,76,79,22, 2172
59  2400 DATA 32,79,22,176,229,2
    01,65,144,227,201,71,176,223
    ,144,219,134, 2343
18  2410 DATA 20,132,21,160,0,17
    7,20,32,177,33,200,201,13,20
    8,246,96, 1736
5A  2420 DATA 162,42,160,8,76,11
    2,22,201,10,176,3,9,48,96,10
    5,54, 1284
C7  2430 DATA 96,72,74,74,74,74,
    32,136,22,32,177,33,104,41,1
    5,32, 1088
15  2440 DATA 136,22,76,177,33,1
    73,119,9,32,146,22,173,118,9
    ,32,146, 1423
F1  2450 DATA 22,169,32,32,177,3
    3,173,121,9,32,146,22,173,12
    0,9,76, 1346
1B  2460 DATA 146,22,32,166,22,1
    62,1,32,29,34,32,221,33,173,
    40,8, 1153
62  2470 DATA 174,41,8,133,20,13
    4,21,32,38,34,162,1,32,29,34
    ,76, 969
DF  2480 DATA 129,22,169,18,32,1
    77,33,32,195,22,169,146,76,1
    77,33,173, 1603
68  2490 DATA 94,9,208,2,240,79,
    169,42,162,8,133,20,134,21,1
    74,102, 1597
60  2100 DATA 9,72,169,0,141,105
    ,9,32,50,22,104,141,105,9,32
    ,204, 1204
81  2110 DATA 255,32,243,20,169,
    15,76,195,255,162,9,160,16,3
    2,50,22, 1711
03  2120 DATA 169,13,76,177,33,3
    2,96,165,134,122,132,123,32,
    115,0,170, 1589
8D  2130 DATA 240,243,162,255,13
    4,58,144,6,32,121,165,76,225
    ,167,32,107, 2167
8F  2140 DATA 169,166,122,160,4,
    189,0,2,232,200,153,251,1,20
    1,0,208, 2058
3E  2150 DATA 244,76,162,164,169
    ,0,141,112,9,141,105,9,141,9
    4,9,141, 1717
C5  2160 DATA 93,9,141,58,9,32,2
    49,18,32,228,18,32,204,255,3
    2,231, 1641
05  2170 DATA 255,96,162,15,32,1
    98,255,32,228,255,32,210,255
    ,201,13,208, 2447
20  2180 DATA 246,76,204,255,174
    ,113,9,208,3,174,114,9,32,38
    ,20,32, 1707
C8  2190 DATA 243,20,169,15,76,1
    95,255,174,113,9,208,3,174,1
    14,9,169, 1946
17  2200 DATA 2,160,0,32,186,255
    ,169,1,162,133,160,9,32,189,
    255,32, 1777
7D  2210 DATA 192,255,176,56,32,
    183,255,208,51,32,117,21,176
    ,46,32,117, 1949
E7  2220 DATA 21,176,36,32,117,2
    1,133,20,134,21,32,45,34,162
    ,1,32, 1017
BC  2230 DATA 29,34,32,135,21,20
    1,0,240,6,32,177,33,76,83,21
    ,169, 1289
0F  2240 DATA 13,32,177,33,76,63
    ,21,169,2,76,195,255,32,104,
    21,162, 1431
45  2250 DATA 21,76,121,17,32,13
    5,21,176,12,141,117,9,32,135
    ,21,176, 1242
3A  2260 DATA 4,170,173,117,9,96
    ,162,2,32,198,255,32,228,255
    ,168,32, 1933
6B  2270 DATA 183,255,8,32,204,2
    55,152,40,208,2,24,96,56,96,
    32,179, 1822
6B  2280 DATA 21,32,195,21,141,1
    13,9,96,32,179,21,32,195,21,
    141,114, 1363
56  2290 DATA 9,96,165,122,166,1
    23,133,251,134,252,32,62,26,
    165,4,208, 1948
F0  2300 DATA 12,96,165,3,201,8,
    144,5,201,12,176,1,96,162,20
    ,76, 1378
70  2500 DATA 9,32,198,255,32,22
    8,255,141,38,8,32,228,255,14
    1,39,8, 1899
69  2510 DATA 208,10,173,38,8,20
    8,5,32,204,255,56,96,32,228,
    255,141, 1949
4C  2520 DATA 40,8,32,228,255,14
    1,41,8,32,228,255,201,0,240,
    13,160, 1882
80  2530 DATA 0,145,20,230,20,20
    8,241,230,21,76,41,23,169,13
    ,145,20, 1602
8F  2540 DATA 32,204,255,24,96,1
    60,0,177,73,153,38,8,200,192
    ,4,144, 1760
80  2550 DATA 246,173,38,8,208,7
    ,173,39,8,208,2,56,96,177,73
    ,240, 1752
7E  2560 DATA 6,153,38,8,200,208
    ,246,169,13,153,38,8,173,38,
    8,174, 1633
95  2570 DATA 39,8,133,73,134,74
    ,24,96,173,28,8,240,1,96,162
    ,9, 1298
84  2580 DATA 76,121,17,169,0,13
    3,71,133,72,133,63,133,64,14
    1,118,9, 1453
53  2590 DATA 141,119,9,141,120,
    9,141,121,9,96,32,132,23,141
    ,96,9, 1339
82  2600 DATA 141,97,9,141,103,9
    ,141,58,9,141,34,8,141,35,8,
    141, 1216
09  2610 DATA 59,9,169,2,141,102
    ,9,165,45,166,46,133,69,134,
    70,160, 1479
71  2620 DATA 0,152,145,69,162,4
    9,160,10,32,50,22,32,208,23,
    96,165, 1375
58  2630 DATA 1,41,254,133,1,96,
    165,1,9,1,133,1,96,169,0,141
    , 1242
36  2640 DATA 234,8,160,0,32,207
    ,255,201,13,240,6,153,235,8,
    200,208, 2160
34  2650 DATA 243,140,234,8,96,1
    62,213,160,9,32,50,22,32,222
    ,23,173, 1819
53  2660 DATA 234,8,208,10,162,2
    37,160,9,32,50,22,76,189,17,
    201,13, 1628
F7  2670 DATA 144,10,162,11,160,
    10,32,50,22,76,246,23,160,0,
    185,154, 1445
F3  2680 DATA 9,153,202,8,200,19
    2,3,208,245,162,0,189,235,8,
    157,170, 2141
B8  2690 DATA 8,153,202,8,232,20
    0,236,234,8,208,240,162,0,18
    9,142,9, 2231
F6  2700 DATA 153,202,8,232,200,
    224,8,208,244,140,37,8,174,2
    34,8,160, 2240
57  2710 DATA 0,185,150,9,157,17
    0,8,200,232,192,4,208,244,14
    2,36,8, 1945
08  2720 DATA 96,173,94,9,240,2,
    208,9,165,43,166,44,133,73,1
    34,74, 1663
08  2730 DATA 96,173,102,9,168,1
    74,113,9,32,186,255,173,36,8
    ,162,170, 1866
61  2740 DATA 160,8,32,80,33,32,
    189,255,32,192,255,176,90,17
    4,102,9, 1819
6C  2750 DATA 32,198,255,176,82,
    32,228,255,32,183,255,208,74
    ,32,228,255, 2525
71  2760 DATA 32,183,255,208,66,
    32,204,255,96,173,93,9,240,5
    6,169,3, 2074
80  2770 DATA 168,174,114,9,32,1
    86,255,173,37,8,162,202,160,
    8,32,189, 1909
4A  2780 DATA 255,32,192,255,176
    ,33,162,3,32,201,255,176,26,
    173,34,8, 2013
00  2790 DATA 32,210,255,32,183,
    255,208,15,173,35,8,32,210,2
    55,32,183, 2118
32  2800 DATA 255,208,4,32,204,2
    55,96,32,204,255,169,16,76,1
    51,17,173, 2147
B5  2810 DATA 27,8,9,48,141,151,
    10,162,134,160,10,32,50,22,7
    6,88, 1128
09  2820 DATA 33,230,251,208,2,2
    30,252,140,23,8,160,0,177,25
    1,172,23, 2160
E4  2830 DATA 8,96,198,251,165,2
    51,201,255,208,2,198,252,96,
    32,19,25, 2257
E8  2840 DATA 169,0,170,141,234,
    8,32,2,25,32,89,22,176,4,142
    ,234, 1480
1C  2850 DATA 8,96,157,235,8,232
    ,208,238,32,2,25,32,8,25,201
    ,32, 1539
EB  2860 DATA 240,246,96,165,45,
    166,46,133,5,134,6,160,0,177
    ,5,208, 1832
1D  2870 DATA 2,24,96,205,234,8,
    208,40,162,0,200,200,200,200
    ,177,5, 1961
2B  2880 DATA 221,235,8,208,27,2
    32,236,234,8,144,242,160,1,1
    77,5,141, 2279
58  2890 DATA 26,8,200,177,5,141
    ,24,8,200,177,5,141,25,8,56,
    96, 1297
C5  2900 DATA 160,0,177,5,24,105
    ,4,101,5,133,5,144,2,230,6,7
    6, 1177
96  2910 DATA 76,25,32,2,25,32,3
    0,25,173,234,8,208,5,162,4,7
    6, 1117
77  2920 DATA 121,17,32,68,25,14
    4,12,162,2,173,27,8,201,1,20
    8,5, 1206
FB  2930 DATA 76,121,17,162,0,16
    9,1,141,28,8,224,0,240,1,96,
    160, 1444
FA  2940 DATA 0,173,234,8,145,69
    ,200,169,2,145,69,200,165,71
    ,145,69, 1864
AE  2950 DATA 200,165,72,145,69,
    200,162,0,189,235,8,145,69,2
    32,200,236, 2327
4E  2960 DATA 234,8,208,244,152,
    24,101,69,133,69,165,70,105,
    0,133,70, 1785
87  2970 DATA 201,192,208,5,162,
    13,76,121,17,160,0,152,145,6
    9,96,169, 1786
13  2980 DATA 0,133,3,133,4,32,2
    ,25,32,97,22,176,1,96,201,65
    , 1022
```

```
51  2990 DATA 176,4,41,15,144,2,
    233,55,24,6,3,38,4,176,25,6,
     952
5A  3000 DATA 3,38,4,176,19,6,3,
    38,4,176,13,6,3,38,4,176, 70
    7
7C  3010 DATA 7,5,3,133,3,76,6,2
    6,162,10,76,121,17,169,0,133
    , 947
11  3020 DATA 3,133,4,32,19,25,3
    2,2,25,32,79,22,176,1,96,41,
     722
05  3030 DATA 15,72,165,4,72,165
    ,3,72,6,3,38,4,176,218,6,3,
    1022
D6  3040 DATA 38,4,176,212,24,10
    4,101,3,133,3,104,101,4,133,
    4,176, 1320
18  3050 DATA 199,6,3,38,4,176,1
    93,104,24,101,3,133,3,165,4,
    105, 1261
37  3060 DATA 0,133,4,176,179,14
    4,191,32,2,25,133,3,169,0,13
    3,4, 1328
C4  3070 DATA 32,2,25,201,39,240
    ,5,162,8,76,121,17,76,2,25,7
    6, 1107
D7  3080 DATA 183,26,173,29,8,24
    0,36,201,60,240,4,165,4,133,
    3,169, 1674
F6  3090 DATA 0,133,4,76,204,26,
    32,8,25,201,13,240,229,201,4
    0,240, 1672
C2  3100 DATA 225,201,41,240,221
    ,201,44,240,217,208,15,169,1
    ,141,31,8, 2203
35  3110 DATA 165,4,240,5,169,2,
    141,31,8,96,201,43,240,4,201
    ,45, 1595
18  3120 DATA 208,72,173,30,8,24
    0,5,162,9,76,121,17,238,30,8
    ,165, 1562
A1  3130 DATA 4,72,165,3,72,32,8
    ,25,72,32,2,25,32,160,26,104
    , 834
AA  3140 DATA 201,43,240,24,166,
    3,164,4,104,133,3,104,133,4,
    56,138, 1520
BD  3150 DATA 229,3,133,3,152,22
    9,4,133,4,76,160,26,24,104,1
    01,3, 1384
A3  3160 DATA 133,3,104,101,4,13
    3,4,76,160,26,162,0,142,30,8
    ,201, 1287
2D  3170 DATA 60,240,4,201,62,20
    8,6,141,29,8,32,2,25,201,39,
    208, 1466
67  3180 DATA 6,32,136,26,76,160
    ,26,32,67,22,144,54,32,30,25
    ,32, 900
8D  3190 DATA 68,25,176,25,169,1
    ,141,32,8,173,27,8,201,2,208
    ,5, 1269
EC  3200 DATA 162,1,76,121,17,16
    9,2,141,31,8,76,160,26,169,0
    ,173, 1332
5E  3210 DATA 24,8,133,3,173,25,
    8,133,4,173,26,8,141,31,8,76
    , 974
86  3220 DATA 160,26,201,36,240,
    4,201,38,208,6,32,0,26,76,16
    0,26, 1440
5E  3230 DATA 32,79,22,144,6,32,
    62,26,76,160,26,162,5,76,121
    ,17, 1046
21  3240 DATA 230,71,208,2,230,7
    2,230,63,208,2,230,64,96,173
    ,93,9, 1981
C5  3250 DATA 240,5,162,3,32,201
    ,255,173,33,8,32,158,34,32,2
    35,27, 1630
3F  3260 DATA 32,161,27,172,31,8
    ,240,25,165,3,32,158,34,32,2
    35,27, 1382
90  3270 DATA 32,161,27,136,240,
    11,165,4,32,158,34,32,235,27
    ,32,161, 1487
BC  3280 DATA 27,173,93,9,240,3,
    32,204,255,96,174,27,8,224,2
    ,240, 1807
94  3290 DATA 1,96,174,93,9,240,
    12,32,210,255,32,183,255,240
    ,3,76, 1911
45  3300 DATA 232,24,96,140,23,8
    ,160,0,145,63,172,23,8,96,14
    1,33, 1364
48  3310 DATA 8,169,0,141,31,8,7
    6,174,27,141,33,8,32,160,26,
    173, 1207
C0  3320 DATA 27,8,201,1,240,58,
    173,31,8,201,1,240,51,165,71
    ,24, 1500
14  3330 DATA 105,2,170,165,72,1
    05,0,168,165,3,134,3,166,4,1
    32,4, 1398
36  3340 DATA 56,229,3,133,3,138
    ,229,4,133,4,201,255,240,12,
    201,0, 1841
D6  3350 DATA 208,22,165,3,201,1
    27,176,16,144,6,165,3,201,12
    8,144,8, 1717
4A  3360 DATA 169,1,141,31,8,76,
    174,27,162,14,76,121,17,141,
    33,8, 1199
17  3370 DATA 24,105,164,133,253
    ,169,13,105,0,133,254,32,8,2
    5,201,65, 1684
20  3380 DATA 208,24,160,1,177,2
    51,201,13,240,4,201,32,208,1
    2,32,2, 1766
B3  3390 DATA 25,169,0,141,31,8,
    168,76,138,29,32,8,25,201,35
    ,208, 1294
A9  3400 DATA 33,32,2,25,32,160,
    26,173,32,8,240,5,169,1,141,
    31, 1110
6E  3410 DATA 8,173,31,8,201,1,2
    40,5,162,10,76,121,17,160,1,
    76, 1290
1D  3420 DATA 138,29,201,40,240,
    81,32,160,26,173,32,8,240,5,
    169,2, 1576
0B  3430 DATA 141,31,8,173,31,8,
    201,1,208,2,160,2,201,2,208,
    2, 1379
13  3440 DATA 160,5,173,33,8,201
    ,110,240,4,201,121,208,2,160
    ,5,192, 1823
34  3450 DATA 5,208,5,169,2,141,
    31,8,32,8,25,201,44,240,3,76
    , 1198
40  3460 DATA 138,29,32,2,25,201
    ,88,240,10,201,89,240,5,162,
    12,76, 1550
C3  3470 DATA 121,17,200,200,76,
    138,29,32,2,25,32,160,26,32,
    8,25, 1123
84  3480 DATA 201,41,240,53,173,
    32,8,240,5,169,1,141,31,8,17
    3,31, 1547
8F  3490 DATA 8,201,1,240,5,162,
    10,76,121,17,32,8,25,201,44,
    240, 1391
77  3500 DATA 5,162,9,76,121,17,
    32,2,25,201,88,208,244,32,2,
    25, 1249
D5  3510 DATA 201,41,208,237,160
    ,8,76,138,29,32,2,25,201,44,
    240,10, 1652
17  3520 DATA 169,2,141,31,8,160
    ,10,76,138,29,173,32,8,240,5
    ,169, 1391
CB  3530 DATA 1,141,31,8,173,31,
    8,201,1,240,5,162,10,76,121,
    17, 1226
F7  3540 DATA 32,2,25,201,89,208
    ,186,160,9,177,253,201,255,2
    40,6,141, 2185
C3  3550 DATA 33,8,76,174,27,162
    ,11,76,121,17,224,56,144,3,7
    6,94, 1302
1B  3560 DATA 30,224,33,144,3,76
    ,110,28,224,25,144,3,76,26,2
    8,76, 1250
96  3570 DATA 15,28,32,132,23,32
    ,240,24,186,142,21,8,32,225,
    255,208, 1603
1E  3580 DATA 3,76,144,33,32,240
    ,22,8,169,42,162,8,133,251,1
    34,252, 1709
B2  3590 DATA 32,19,25,40,144,31
    ,173,103,9,208,1,96,173,102,
    9,32, 1197
D6  3600 DATA 195,255,169,0,141,
    103,9,169,2,141,102,9,173,10
    4,9,141, 1722
6B  3610 DATA 94,9,76,52,30,169,
    0,141,28,8,141,29,8,141,30,8
    , 964
72  3620 DATA 141,32,8,32,134,34
    ,32,57,25,201,59,240,38,201,
    46,208, 1488
A8  3630 DATA 6,32,147,25,32,60,
    25,201,13,240,24,32,67,22,17
    6,5, 1107
CE  3640 DATA 162,6,76,121,17,32
    ,30,25,32,32,22,72,32,60,25,
    104, 848
C1  3650 DATA 32,155,29,173,27,8
    ,201,2,208,11,173,58,9,240,3
    ,32, 1361
9F  3660 DATA 195,22,32,73,30,76
    ,189,29,165,71,166,72,141,11
    8,9,142, 1530
14  3670 DATA 119,9,165,63,166,6
    4,141,120,9,142,121,9,96,224
    ,56,208, 1712
40  3680 DATA 3,76,24,31,224,57,
    208,3,76,97,31,224,58,208,3,
    76, 1399
63  3690 DATA 129,31,224,59,208,
    3,76,173,31,224,60,208,3,76,
    252,30, 1787
49  3700 DATA 224,61,208,3,76,17
    2,30,224,62,208,3,76,160,30,
    224,63, 1824
43  3710 DATA 208,3,76,166,30,22
    4,64,208,3,76,231,32,76,119,
    33,169, 1718
8C  3720 DATA 1,141,58,9,96,169,
    1,141,59,9,96,32,60,25,32,16
    0, 1089
8E  3730 DATA 26,173,93,9,240,5,
    162,3,32,201,255,169,0,166,3
    ,164, 1701
2F  3740 DATA 4,202,224,255,208,
    14,136,192,255,208,9,173,93,
    9,240,3, 2225
A9  3750 DATA 32,204,255,96,32,2
    22,30,32,161,27,76,194,30,17
    3,27,8, 1599
C9  3760 DATA 201,2,240,1,96,173
    ,93,9,240,5,169,0,76,210,255
    ,140, 1910
C2  3770 DATA 23,8,160,0,152,145
    ,63,172,23,8,96,32,60,25,32,
    160, 1159
0E  3780 DATA 26,165,3,166,4,133
    ,71,134,72,133,63,134,64,141
    ,34,8, 1351
61  3790 DATA 142,35,8,96,32,57,
    25,32,60,25,201,39,240,35,32
    ,160, 1219
22  3800 DATA 26,165,4,240,5,162
    ,10,76,121,17,32,37,32,32,16
    1,27, 1147
92  3810 DATA 32,60,25,201,44,24
    0,221,201,13,208,1,96,162,9,
    76,121, 1710
91  3820 DATA 17,32,2,25,201,0,2
    40,244,201,39,240,11,133,3,3
    2,37, 1457
85  3830 DATA 32,32,161,27,76,66
    ,31,32,2,25,76,49,31,32,57,2
    5, 754
72  3840 DATA 32,60,25,32,160,26
    ,32,45,32,32,161,27,32,161,2
    7,32, 916
25  3850 DATA 60,25,201,44,240,2
    31,201,13,208,1,96,162,9,76,
    121,17, 1705
79  3860 DATA 32,121,23,165,6,72
    ,165,5,72,32,160,26,104,133,
    5,104, 1225
E7  3870 DATA 133,6,165,4,240,5,
    162,10,76,121,17,160,1,169,1
    ,145, 1415
```

```
68   3880 DATA 5,200,165,3,145,5,
     200,165,4,145,5,96,32,121,23
     ,165, 1479
5E   3890 DATA 5,72,165,6,72,32,1
     60,26,104,133,6,104,133,5,16
     0,1, 1184
60   3900 DATA 169,2,76,160,31,18
     6,142,20,8,32,155,23,169,1,1
     41,112, 1427
48   3910 DATA 9,32,154,33,173,93
     ,9,13,94,9,240,3,32,246,23,1
     69, 1332
75   3920 DATA 1,141,27,8,32,98,2
     4,32,179,29,32,14,18,238,27,
     8, 908
52   3930 DATA 32,98,24,32,170,24
     ,32,179,29,32,3,32,32,127,32
     ,76, 954
DE   3940 DATA 189,17,162,162,160
     ,10,32,50,22,173,35,8,32,146
     ,22,173, 1393
E6   3950 DATA 34,8,32,146,22,162
     ,199,160,10,32,50,22,32,166,
     22,169, 1266
27   3960 DATA 13,76,177,33,169,1
     ,141,31,8,76,50,32,169,2,141
     ,31, 1150
27   3970 DATA 8,173,93,9,240,40,
     208,0,173,27,8,201,2,240,1,9
     6, 1519
83   3980 DATA 162,3,32,201,255,1
     72,31,8,165,3,32,158,34,32,2
     10,255, 1753
CA   3990 DATA 136,240,8,165,4,32
     ,158,34,32,210,255,76,204,25
     5,173,27, 2009
3D   4000 DATA 8,201,2,240,1,96,1
     74,31,8,165,3,32,158,34,160,
     0, 1313
01   4010 DATA 145,63,200,202,240
     ,7,165,4,32,158,34,145,63,96
     ,173,59, 1786
95   4020 DATA 9,208,1,96,162,217
     ,160,10,32,50,22,165,45,166,
     46,133, 1522
E8   4030 DATA 5,134,6,160,0,177,
     5,208,1,96,141,23,8,162,0,20
     0, 1326
F4   4040 DATA 200,200,200,177,5,
     32,177,33,232,236,23,8,208,2
     44,169,32, 2176
CF   4050 DATA 32,177,33,169,36,3
     2,177,33,160,1,177,5,160,3,2
     01,1, 1397
12   4060 DATA 240,5,177,5,32,146
     ,22,136,177,5,32,146,22,160,
     0,177, 1482
E4   4070 DATA 5,24,105,4,101,5,1
     33,5,165,6,105,0,133,6,169,1
     3, 979
04   4080 DATA 32,177,33,76,148,3
     2,173,103,9,240,5,162,19,76,
     121,17, 1423
F0   4090 DATA 32,60,25,201,34,20
     8,70,162,0,32,2,25,201,34,24
     0,10, 1336
A5   4100 DATA 157,60,9,232,224,1
     6,144,241,176,54,142,92,9,16
     9,4,141, 1870
B4   4110 DATA 102,9,141,103,9,17
     3,94,9,141,104,9,169,1,141,9
     4,9, 1308
27   4120 DATA 173,102,9,168,174,
     113,9,32,186,255,173,92,9,16
     2,60,160, 1877
BC   4130 DATA 9,32,80,33,32,189,
     255,32,88,33,76,137,24,162,1
     5,44, 1241
6A   4140 DATA 162,16,169,2,141,1
     02,9,169,0,141,103,9,76,121,
     17,141, 1378
96   4150 DATA 23,8,134,3,132,4,9
     6,173,94,9,208,1,96,162,234,
     160, 1537
EC   4160 DATA 10,32,50,22,160,0,
     177,3,32,177,33,200,204,23,8
     ,208, 1339
A0   4170 DATA 245,169,13,76,177,
     33,173,93,9,240,5,162,18,76,
     121,17, 1627
BC   4180 DATA 32,60,25,32,160,26
     ,165,3,166,4,133,63,134,64,9
     6,162, 1325
C8   4190 DATA 119,160,11,32,50,2
     2,76,189,17,173,93,9,240,1,9
     6,160, 1448
E4   4200 DATA 1,177,43,240,1,96,
     162,158,160,11,32,50,22,76,1
     89,17, 1435
F0   4210 DATA 141,106,9,173,105,
     9,240,30,142,107,9,140,108,9
     ,162,7, 1497
0A   4220 DATA 32,201,255,173,106
     ,9,32,210,255,32,204,255,173
     ,106,9,174, 2226
9B   4230 DATA 107,9,172,108,9,96
     ,173,106,9,76,210,255,173,11
     5,9,240, 1867
A1   4240 DATA 44,206,115,9,173,1
     09,9,32,146,22,162,1,32,29,3
     4,172, 1295
B2   4250 DATA 115,9,240,30,173,1
     10,9,32,146,22,162,1,32,29,3
     4,136, 1280
5E   4260 DATA 240,21,173,111,9,3
     2,146,22,162,1,76,29,34,162,
     3,32, 1253
AF   4270 DATA 29,34,162,3,32,29,
     34,162,3,76,29,34,169,32,32,
     177, 1037
BE   4280 DATA 33,202,208,250,96,
     162,0,142,122,9,240,9,162,1,
     142,122, 1900
9D   4290 DATA 9,202,142,123,9,16
     0,0,56,200,165,20,253,125,9,
     133,20, 1626
C8   4300 DATA 165,21,253,126,9,1
     33,21,176,239,136,165,20,125
     ,125,9,133, 1856
F4   4310 DATA 20,165,21,125,126,
     9,133,21,152,141,124,5,201,0
     ,208,13, 1468
70   4320 DATA 173,122,9,240,5,17
     3,123,9,240,13,173,124,9,9,4
     8,32, 1502
1A   4330 DATA 177,33,169,1,141,1
     23,9,232,232,224,8,208,184,1
     65,20,9, 1935
AA   4340 DATA 48,32,177,33,96,16
     9,234,141,33,8,133,3,133,4,1
     41,109, 1494
89   4350 DATA 9,141,110,9,141,11
     1,9,169,0,141,115,9,96,142,2
     2,8, 1232
E1   4360 DATA 174,115,9,224,3,24
     0,7,157,109,9,232,142,115,9,
     174,22, 1741
CA   4370 DATA 8,96,0,0,0,0,0,0,0
     ,0,0,0,0,0,0,0, 104
```



Extending Basic

Declaring and using labels in Basic is not as difficult as it may seem

By Burghard-Henry Lehmann

One advantage of writing machine code programs with an assembler is that you can declare labels for jumps and branches. An example of this is "Speedy Assembler", written by yours truly, and still available from Readers' Services for everybody who wants to get into serious machine code programming!

Instead of having to give jumps and subroutine calls as absolute addresses, and having to calculate relative branches, you simply declare a label on the left-hand side at the start of the routine to be jumped or branched to, and the assembler does the rest. It stores the label in what is called "the symbol table", with the address of the location at that point next to it.

Later on, when it finds that label next to a jump or branch instruction, it fetches the address from the symbol table and, in case of a jump or subroutine call, assembles it as the location to be jumped or to be called or, in the case of a branch instruction, calculates the length of the branch. The programmer doesn't have to trouble himself with any of this. Nor does he have to go to the trouble of calculating branches. Instead he just attaches a name to the routine or subroutine in question.

And what's more, labels like this add tremendously to the readability of the program, because you can give each routine and subroutine a name that suits you. This helps quite a bit when a bug has developed in the program (and doesn't it always!), and the programmer has to spend ages finding it!

Labels in Basic

Because of all this, I felt for a long time that it would be nice to be able to use labels like this in Basic programs too. No more memorizing of line numbers, and, most of all, no more renumbering of GOTO's and GOSUB's whenever one changes the program.

It is actually surprisingly easy to introduce such a facility to the rather poverty-stricken Commodore 64 Basic. In the last article of this series we've developed a routine which allows us to give GOTO and GOSUB with variable names. So we've already got the basic facility to labelize jumps and subroutine calls, but we still have to declare the value of the label at the beginning of the program with a line like: "Subroutine = 1000". To do this job for us, the computer has to build a symbol table of sorts, and this has to be done before the program is actually run. This is because, during execution, when a GOTO or GOSUB is encountered, the computer has to know where to jump to.

You may know that most assemblers are called "Two Pass Assemblers". This is because in order to deal with jumps and branches, the assembler has to do its job in two gos. First it goes through the whole of the textfile and builds the symbol table, and then it has a second go in which it is able to assemble the source in earnest.

To use labels to the full in Basic, we have to do a similar thing. Before the program is run in earnest, the computer has to sift through the whole of the textfile and collect all the line numbers to which it has to GOTO and GOSUB later on. This means that it

Figure 1

```
A871   BasicRun   PHP          Save status register
A872              LDA   0      This is direct run mode
A874              JSR $FF90    Set Kernal flags to direct run mode
A877              PLP          Retrieve status register
A878              BNE LineNo   If Z=0 then RUN plus line number
A87A              JMP          Do CLR and start program

A87D   LineNo     JSR $A660    Do CLR
A880              JMP $A897    Jump to GOTO and starT program
```

takes a bit of time until running proper starts. But I don't think that this is a major handicap, and it certainly is worth it!

Running a Basic Program

So we have to intercept the RUN command. To do this, let's look first of all a bit closer at what it does under normal circumstances.

After you've typed in your Basic program and then given the RUN command to execute your program, the Basic interpreter jumps to the routine at $A871. Figure 1 gives you a disassembly of that routine.

First the flag register is saved on the stack. Then a call to a Kernal routine is made. This routine puts the computer into the direct run mode by loading the system variable $9D with zero and setting ST, which is the status variable.

Then the flag register is pulled from the stack again. If the zero flag equals one, that is, if the last value in the accumulator has been zero, then there were no parameters with the RUN command. In this case the routine continues, otherwise it branches forward, because a line number has been given with RUN.

If no line number is given with RUN the routine jumps to the CLR routine and doesn't return. If a line number has been given with RUN, the routine calls on the CLR routine and then jumps to the GOTO routine, because in the end, RUN 100 is similar to GOTO 100. The only difference is that RUN clears all the Basic variables, while GOTO leaves them untouched.

The CLR routine clears all the Basic variables and gives Basic a fresh start, so that variables and arrays can be built up anew. This is mainly done by setting the string storage pointer and the array storage pointer to the end of the Basic textfile, because, as you might know, Basic stores all the variables and arrays declared in a program directly after the end of the textfile.

Modified RUN

Our modified RUN routine starts at line 1520 (Listing 1). First we deal with the RUN command more or less in the same way as the ROM routine. That is, we set the Kernal flags to the direct run mode (line 1520-1530), and then we clear the Basic variables (line 1540).

I haven't bothered about RUN line number, so this won't work with the routine as it is. If you give a line number with RUN, it will just be ignored. But nevertheless, if you do want this little-used facility, it shouldn't be impossible for you to add it with the help of the explanation of the normal RUN routine, which I've given above.

Next, we set zero page 254/255 to the start of the Basic textfile (line 1580-1610). Zero page locations 251 to 255 are never used by the Commodore operating system, so these locations are absolutely safe to be used for your own purposes. There are many other zero page locations which are used by the Kernal or the Basic interpreter, but which are usually quite safe to use. For example, if you don't do any floating point arithmetic, you may use locations $61 to $6F without any trouble. But the point is always to think before you use a zero page variable, otherwise the system might do very funny things indeed!

After that we go into the main loop (SEARCHLP), whch looks at each Basic line to see if it contains a label (line 1630-1810). To understand SEARCHLP, here is a short explanation of how a Basic line is stored in memory:

First of all the line number is given in the usual low byte/high byte fashion. Next there are two bytes which contain the so-called link pointer. Each link pointer points to the beginning of the next line. This makes it very easy to search through a Basic textfile, because you just have to jump from one link pointer to the other, each time looking at the line number preceding it, and in next to no time you've found the line you're looking for.

Each Basic line is finished with a zero, which is the standard terminator used by Commodore (strings too are always terminated with zero!). At the end of the textfile there will be two zeros in the locations where otherwise the next line number would be. So this is how the computer will know when it has reached the end of the textfile.

If you look at SEARCHLP in our program, you'll find the routine testing for those two zeros right at the beginning of the loop (line 1630-1680).

Declaring a Label

Naturally we have to tell the computer when it has found a label. For that we have to make a label stand out in some way.

To do this I have chosen the following way of declaring a label: a label has to be at the beginning of a line, after the line number, and it has to be preceded by a full stop. Of course, you are free to experiment with methods which might suit you better, because this is the whole purpose of this series of articles, to enable you to develop extended Basic routines which suit your particular needs!

Anyhow, in the routine given, the computer looks for a full stop and that tells him that it has found a label (line 1690-1720). So it jumps to the routine which I called LABLFOUND. First of all the current location in the textfile is saved in $7A/7B, so that afterwards the computer can continue searching the texFile for more labels (line 1960-1990).

Then it goes forward five bytes to point at the label itself (line 2000-2060). Remember, all we are doing here is a simple LET operation, like, "LET LABEL = 1000" (or "LABEL = 1000" if you omit the LET). So the next ROM routine ($BO8B) we call validates our label, that is, finds out if it is a permitted variable name (as you know, a valid Basic variable name has to start with a letter).

If the name is valid, the first two characters of it are stored in the variable area, which starts immediately after the end of the Basic textfile. On return from $B08B, the low byte of the variable location is in the accumulator, and the high byte is in the Y-Register. We store this in zero page $49/4A which is the system variable pointer (line 2110-2120).

Next we get the line number and store it in zero page $62/63, which is the floating point accumulator # 1 (line 2160-2210). The next routine ($B399) which we call converts the line number into a proper floating point number (line 2250). We have to do it in this rather roundabout way because variables of this type have to be stored as floating point numbers in order to be recognized later on!

Finally we call $BBD0 (line 2260), which stores the value contained in flp accu # 1 in the variable area itself. Now our label has been stored like any other Basic variable in the variable store, including the number of the line on which it appeared. We can return to our main search loop and look for the next label.

The fact that each of the labels we

declare is stored like any other Basic variable means, of course, that only the first two characters of the label will be recognized. The rest of the label is ignored, which limits the use of labels rather, because it doesn't give much scope for declaring meaningful names which are recognized.

Therefore, in the next article I'd like to develop a routine which builds and recognizes a symbol table away from the Basic variable area. This will enable us to declare labels with six or more characters which are fully recognized.

Listing 1

```
            ORG 49152
            ENT
;
CHARGET     EQU $0073
EXECVECT    EQU $0308
;
;
;
;TURN EXTENDED BASIC ON
;BY CHANGING VECTOR AT $0308
;
EXTBASON    LDA #<PRGSTART
            STA <EXECVECT
            LDA #>PRGSTART
            STA >EXECVECT
;
            RTS
;
;
;
;TURN EXTENDED BASIC OFF
;BY CHANGING VECTOR AT $0308
;BACK TO NORMAL ($A7E4)
;
EXTBASOFF   LDA #<$A7E4
            STA <EXECVECT
            LDA #>$A7E4
            STA >EXECVECT
;
            RTS
;
;
;
;*** MAIN PROGRAM ENTRY ***
;
;LOOK FOR EXTENDED BASIC COMMANDS
;
PRGSTART    JSR CHARGET
            JSR EXECSTM
            JMP $A7AE
;
EXECSTM     CMP 'O
            BNE NEXT
            JMP OFF.RT
NEXT        CMP #$89        ;'GOTO'
            BEQ GOTO.RT
            CMP #$8A        ;'RUN'
            BNE NEXT1
            JMP RUN.RT
NEXT1       CMP #$8D        ;'GOSUB'
            BEQ GOSUB.RT
            CMP '.          ;SPECIAL
            BEQ SPEC.RT
;
            CMP 'C
            BNE NORMAL
            JSR CHARGET
            CMP 'O
            BNE NORMAL
            JSR CHARGET
            CMP 'L
            BNE NORMAL
            JSR CHARGET
            CMP #$B0   ;'OR' TOKEN
            BEQ COLOR.RT
;
;DO NORMAL ROM-ROUTINE
;
NORMAL      JMP $A7ED
;
;
;EXECUTE 'COLOR' COMMAND
;
;GET INK PARAMETER
;
COLOR.RT    JSR CHARGET
            JSR $AD8A
            JSR $B7F7
;
;CHANGE INK COLOUR
;
            STY 646
;
;GET PAPER PARAMETER
;
            JSR CHARGET
            JSR $AD8A
            JSR $B7F7
;
;CHANGE PAPER COLOUR
;
            STY 53281
;
;GET BORDER PARAMETER
;
            JSR CHARGET
            JSR $AD8A
            JSR $B7F7
;
;CHANGE BORDER COLOUR
;
            STY 53280
;
;JUMP TO REST OF ROM-ROUTINE
;
            RTS
;
;
;EXECUTE 'GOTO' COMMAND
;
GOTO.RT     JSR CHARGET
GOTO.RT1    JSR $AD8A
            JSR $B7F7
            JSR $A8A3
            RTS
;
;
;EXECUTE 'GOSUB' COMMAND
;
GOSUB.RT    JSR CHARGET
            LDA #$03
            JSR $A3FB
            LDA $7B
            PHA
            LDA $7A
            PHA
```

```
1290            LDA $3A
1300            PHA
1310            LDA $39
1320            PHA
1330            LDA #$8D
1340            PHA
1350            JSR GOTO.RT1
1360            JMP $A7AE
1370  ;
1380  ;
1390  ;
1400  ;IGNORE LABELS
1410  ;
1420  SPEC.RT   JSR CHARGET
1430            BNE SPEC.RT
1440  ;
1450            RTS
1460  ;
1470  ;
1480  ;MODIFIED 'RUN'-ROUTINE
1490  ;
1500  ;DO 'RUN'
1510  ;
1520  RUN.RT    LDA #0
1530            JSR $FF90
1540            JSR $A659
1550  ;
1560  ;GO THROUGH PRG & ASSEMBLE LABELS
1570  ;
1580            LDA #<$0801
1590            STA <254
1600            LDA #>$0801
1610            STA >254
1620  ;
1630  SEARCHLP  LDY #0        ;END OF PRG?
1640            LDA (254),Y
1650            BNE SEARCHLP1
1660            INY
1670            LDA (254),Y
1680            BEQ EXIT
1690  SEARCHLP1 LDY #4               ;'.'?
1700            LDA (254),Y
1710            CMP '.
1720            BEQ LABLFOUND
1730  SEARCHLP2 LDY #0        ;NEXT LINE
1740            LDA (254),Y
1750            PHA
1760            INY
1770            LDA (254),Y
1780            STA >254
1790            PLA
1800            STA <254
1810            JMP SEARCHLP
1820  ;
1830  ;
1840  ;ALL DONE - RESET CHARACTER
1850  ;POINTER AND START PROGRAM
1860  ;
1870  EXIT      JSR $A68E
1880            JMP $A7EA
1890  ;
1900  ;
1910  ;
1920  ;STORE LABEL IN VARIABLES AREA:
1930  ;
1940  ;POINT TO BEGINNING OF LABEL
1950  ;
1960  LABLFOUND LDA 254
1970            STA $7A
1980            LDA 255
1990            STA $7B
2000            CLC
2010            LDA $7A
2020            ADC #5
2030            STA $7A
2040            LDA $7B
2050            ADC #0
2060            STA $7B
2070  ;
2080  ;STORE LABEL
2090  ;
2100            JSR $B08B
2110            STA $49
2120            STY $4A
2130  ;
2140  ;GET LINE NUMBER
2150  ;
2160            LDY #2
2170            LDA (254),Y
2180            STA $63
2190            INY
2200            LDA (254),Y
2210            STA $62
2220  ;
2230  ;AND STORE IN FLP ACCU
2240  ;
2250            JSR $B399
2260  ;
2270  ;STORE VALUE IN VARIABLE STORE
2280  ;
2290            JSR $BBD0
2300  ;
2310  ;FINISHED
2320  ;
2330            JMP SEARCHLP2
2340  ;
2350  ;
2360  ;
2370  ;TEST FOR REST OF 'OFF'
2380  ;
2390  OFF.RT    JSR CHARGET
2400            CMP 'F
2410            BNE NORMAL1
2420            JSR CHARGET
2430            CMP 'F
2440            BNE NORMAL1
2450  ;
2460  ;EXECUTE 'OFF' COMMAND
2470  ;
2480            JSR EXTBASOFF
2490  ;
2500  ;GET NEXT CHARACTER AND
2510  ;JUMP TO REST OF ROM-ROUTINE
2520  ;
2530            JSR CHARGET
2540            RTS
2550  ;
2560  ;
2570  ;
2580  NORMAL1   JMP $A7ED
```

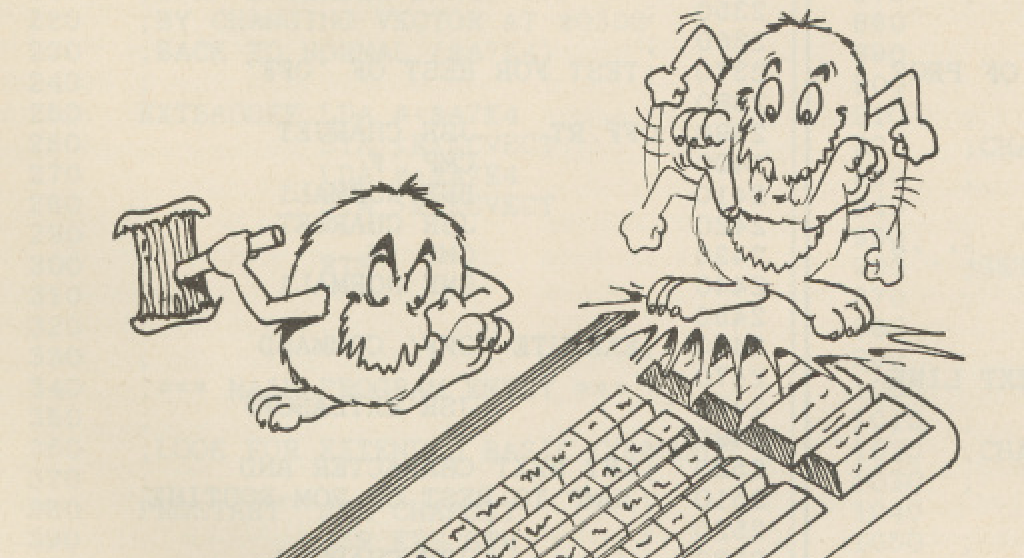
Listing 2

```
110 .BEGINNING
120 INK=INT(RND(1)*16)
130 PRINT "INK=";INK;" ";
140 PAPER=INT(RND(1)*16)
150 PRINT TAB(15) "PAPER=";PAPER;" ";
160 BRD=INT(RND(1)*16)
170 PRINT TAB(28) "BORDER=";BRD
180 COLOR INK,PAPER,BRD
190 GOSUB DELAY
200 GOTO BEGINNING
210 .DELAY
220 FOR N=1 TO 500
230 NEXT
240 RETURN
```

Tech Troubles

A selection of the problems solved and readers' hints from this month's mailbag

By Andy Andros



Why do computers use integers as well as normal decimal values, and where do floating point numbers come into it?
David Kennedy, Easingwold

Dear David,
Another term for an integer is a 'whole number' or, as you point out, a number without any decimal point. Other numbers are referred to as 'floating point' numbers because of the way in which the mathematics is performed in the computer's memory. Floating point calculations are difficult to explain in the space allowed here but, luckily, the fact that they *do* work is all that you really need to know for Basic.

The reason that computers use integers is because it helps to speed-up arithmetic, and is more accurate than floating points where a certain amount of rounding up and down of values occurs. The speed benefit is gained because there is no need to calculate the position of the decimal place, or to check if the value should be expressed in exponential form, where an extremely large or small number is expressed by a value, and the power to which it has to be raised to reveal the actual number.

A popular myth is that integers use less memory than floating point values, but this is not entirely true. Integers use four bytes – two for the variable name, and two for the actual number. Floating point numbers similarly use two bytes for the variable name, but five more bytes are needed for the value. So, on the face of it,

integers do need less space, but the integer value is followed by three unused bytes, which means that both types of variable physically occupy seven bytes of memory.

I bought the **Your Commodore Serious Users Guide 1988**, *and found the 64 tips for the 64 quite informative. One of the tips is how to simulate a PRINT AT command without using rows of cursor downs and lefts. I thought that you may be interested in a technique that I've been using for the past few years:*

```
10 POKE214,9:PRINT"[UP]"
20 POKE211,15
30 PRINT"HELLO"
```

Location 214 is the vertical position of the cursor on the screen (value 1 to 24). The PRINT command in line 10 is used to update this new row in the Commodore's memory, and a cursor up to replace the cursor onto the required line. Next the cursor is placed horizontally by poking a value to location 211 and the message can then be printed.

Two final words on this routine relate to printing on the top and bottom lines. For using the top line, printing a cursor home with a semicolon will set the row position and then the column value can be poked into 211. If the last line is used the screen will scroll up unless you follow the message printing command with a semicolon.

I hope this is of use to someone, somewhere.
Carol Seddon, Wigan

Dear Carol,
Thanks very much for that tip. I found it very informative, and if anyone else has a technical tip I'd be very pleased to hear about it.

In a listing I came across the following syntax which I don't understand:

```
230 HEX$=CHR$(A+48-(A>9)*7)
```

Could you explain what it achieves?
Clive Fowler, Midhurst

Dear Clive,
This is an unusual but useful application of a comparison command. It is the sort of routine which would convert a decimal value to its hexadecimal equivalent.

To understand how it works, I'll have to explain the principle behind the *greater than* and *less than* commands. When a comparison is made, the result is a zero if the condition is false and a minus one if the condition is true.

In the example that you've supplied, the value given by the comparison would be zero if X had a value between zero and nine but minus one if the value exceeds nine. What the equation is doing is taking the value in A and adding 48 to give a value whose CHR$ equivalent is the number character which corresponds to X. This works well up to a value of nine, but values 10 to 15 would produce punctuation marks and symbols because there is a gap of seven characters between the CHR$ value for nine and the CHR$ value for the letter A. The program has to be able to add seven to values over 10, but not to values below that.

The solution is to compare the value with nine, and then multiply this by seven. This would mean that the low values would subtract zero from the CHR$ value but high values would subtract -7 (-1 multiplied by 7). Subtracting a negative value follows the mathematical law that states that two minuses gives a plus, so the program actually adds seven to the CHR$ value.

To test this try setting up a loop for A having values of zero to 15. Place the equation inside the loop and a statement to print each HEX$ value, and the result will be a list of hex equivalents from O to F.

I have written a program which uses several groups of DATA statements but the only problem with it is that RESTORE can't be set to point to a particular line. Is there any easy way around this, because at the moment I have to RESTORE and then read in DATA until the correct point is reached?
Harry Ivy, Hampton

Dear Harry,
There are two ways to tackle your problem. The first is to read the data into several arrays and manipulate these instead of using data READs.

The second method is to create a special form of the RESTORE command. This is done by reading in the data until the line where the program is to be restored to. The actual memory location can be stored by peeking locations 65 and 66 and storing this in a table in a safe part of memory.

Each suitable point is similarly stored until the table is complete. When the data reading has to be restored to any of these points, they can be peeked from their storage point and poked back to 65 and 66:

Can you tell me where I can find a good book containing a breakdown of the C128 Basic ROMs? I have a hunch that such a book may not exist, because I have a book which gives a breakdown of the Kernal, but which states that the Basic disassembly would fill a very large volume.
Richard Trey, York

Dear Richard,
You'll be pleased to hear that such a book does actually exist, though you are right in assuming that it is a weighty tome. *C128 Basic 7.0 Internals*, published by Abacus, runs to over 600 pages. Of this, 450 pages contain an annotated ROM disassembly, and the rest is packed with useful information about the workings of the system, alongside some useful programming hints and routines. The price is £16.95, and it is available from Precision Software, 6 Park Terrace, Worcester Park, Surrey KT4 7JZ, but please include £1.40 for post and packing.

If you have a problem let us know and Andy will try to help. Write to: Tech Troubles, *Your Commodore*, Argus House, Boundary Way, Hemel Hempstead, Herts HP2 7ST.

THE EPSON SQ-2500

To conclude last month's feature on the world of 24 pin dot matrix printers, Robin Burton admires the highly impressive Epson SQ-2500

The first major task is getting the SQ-2500 out of the box – ideally, it's a job for two people. The SQ-2500 might be expensive, but it has a wide carriage and you certainly get a lot of printer for your money. Styling is recognisably Epson, but the greatest impression from the first is that this printer oozes quality. Construction is superb. I didn't actually try it, but I'm sure I could jump on the SQ-2500 without damaging it (and I weigh over 16 stones).

Setting it up

This is an ink-jet printer so there's no ribbon, just an ink cartridge, paper feed knob, the mains lead and paper guide. Open a cover on the right side towards the rear of the printer, push in the ink cartridge, close the cover and that's done. On the left side is a similar cover, but unless you want to use either of the possible two extra font cartridges, you needn't open it.

The interface and power lead connect at the rear, as usual for Epson, and the power switch is on the right side. The paper guide easily slots into position, and when an new ink cartridge is fitted, the SQ needs priming (this is the long cleaning cycle mentioned later, but it just means switch on and press a button). After about half a minute it's ready to go.

Both parallel and a serial interface are standard, with provision for a third one, and any of the extra Epson interfaces (listed in the LQ-500 report) can be used too. The standard buffer is 8K. The manual is 170 pages, in contrast to the LQ's 230, but contains all the necessary ingredients, including both types of command summary and a quick reference card.

If you can afford an SQ, buy one! If you do, don't waste time looking for dip-switches – there aren't any! Four buttons (and one of those is on/off line) control EVERYTHING!! This explains the short manual. There's very little on trouble-shooting. The only way I can envisage anyone having trouble with the SQ is if they can't read, in which case the manual wouldn't help much.

I'm not being flippant! It's all set-up, and manual selection is by question and answer, but with a difference. There's a liquid crystal display at the front of the printer next to the selection buttons, and the SQ asks the questions. You answer by simply pressing a button, and the settings are stored. This is a dream machine.

Superb though this is, there's more! The SQ also has four macros. Each of these is a complete, permanently stored definition containing everything you could want the printer to know about a job. Any one of the four can be loaded automatically at power-up: you choose which one you want in the configuration defaults, which remain permanently set unless you alter them.

You can manually load any macro by a couple of button presses, and can also amend them manually, by software or both, either temporarily, or permanently. Simply re-save a macro at any time (by pressing a button) if you want the changes to be permanent. If not, they're forgotten, either when you switch off or when you load a different macro. You might also want to print out the settings, in which case – press a button.

This is all so comprehensive yet simple that describing it is difficult. I've therefore included a print of my configuration settings separately, so you can see for yourself. Everything is stored – left and right margins, font, style, pitch, page size, etc, etc.

As standard, the SQ-2500 is a cut sheet machine, but an optional tractor unit can be added for continuous paper. You may have gathered by now that the SQ-2500 is intended for high quality, high speed, high volume output. The fact that the tractor is top-mounted and therefore without paper parking is, in context, irrelevant.

Don't misunderstand – swapping between continuous and sheet-feed is easy enough, but if you need to do it very often you don't need an SQ-2500. It wasn't built for the average home user's mixed low-volume needs, and used that way would be like doing the grocery shopping in a Formula One car, theoretically possible but...

If cut sheet is the major use a double-bin automatic sheet feeder can also be attached.

Specification

The SQ-2500 is of course above all an Epson, so the compatibility comments for the LQ-500 all apply to the SQ, including the 13 national character sets.

Six fonts are standard for the SQ-2500, each as usual with italic. Additional effects are limited to double width, double height or both. All fonts are available in the usual 10, 12 and 15 cpi, and all can be condensed except 15 pitch.

There's also provision for two font modules, but so far as I could see all the Epson LQ fonts are standard in the SQ except OCR-B. Minimum vertical spacing is in 180ths of an inch, and horizontal is in 360ths.

Using the Epson

When the SQ powers-up it automatically goes through a self-cleaning cycle. The printer informs you (via the LCD) when the ink cartridge is getting low, and when it's exhausted. According to the manual, a cartridge lasts for 3,000,000 characters in LQ, and 6,000,000 in draft. This seems fair enough – I've gone through well over 1,500 sheets, largely in LQ, (roughtly 6,000,000 characters) and not a word so far.

Using semi-automatic sheet feed is simple and quick, and it has been absolutely reliable with all weights of paper. Just drop a sheet into the guide, press 'form feed' and the sheet is lined up perfectly (I even tried it with a 9 x 4 inch envelope – notoriously difficult to keep straight – with no trouble at all).

If the tractor is fitted, it simply attaches on top of the SQ – there are no covers to change. Usefully, along with the tractor, Epson provide a matching base on which the printer sits. Continuous paper is kept inside this, out of the way. A paper rest is also included to keep the paper clear of the leads.

Manual line feed, form feed, self-test and hex. dump are controlled by the buttons, and the self test includes the current configuration and macro settings too. As mentioned before, just the configuration and the four macros can also be listed on demand (or alternatively checked directly via the LCD). Without altering any of the current settings you also can switch between draft or LQ at any time by pressing either the line-feed or the form-feed button.

Print quality is frankly so superior to that of an impact dot-matrix printer that a comparison is pointless. Because there are no wires pressing through a fabric ribbon, the individual dots are much smaller and more precise. I've even asked when I bought a laser-printer by people who didn't know I had the SQ. It might not be quite that good, but it's obviously near enough, and you'd have to put the two side by side to notice a difference. Also unlike any impact printer, the quality never varies. There's no deterioriation or fading of print quality, because there's no ribbon to wear out. Characters are always perfect, and uniformly sooty black.

The SQ is the fastest matrix printer I've ever seen. The figures don't adequately tell the story – you have to see it to appreciate just how quick it is. In fact its speed was at one point an inconvenience. With such dense character images, double strike is almost unnecessary. Of course I tried it, but output was quick enough to allow the paper to re-fold in the collection basket before the ink had fully dried. I have now lowered the basket.

The final difference about the SQ is noise, specifically the absence of it – only ink touches the paper. You don't realise just how much you are mentally conditioned to brace yourself for the onslaught of noise from an impact printer, at least not until you start printing and it doesn't happen. It's rather uncanny at first, and after years of impact printers, it took me a week to get used to this. Needless to say the SQ easily passes the telephone test, in fact if you were surrounded by them it would be no problem.

Quite aside from its obvious desirability, this aspect of the SQ's performance (along with the others) offers very serious benefits for offices where several printers are at work. I would think the amount of noise from the SQ (just the carriage moving back and forth) would be acceptable even in places like libraries and hospitals too.

The SQ-2500 has operated perfectly throughout.

Conclusion

It hardly needs saying that the SQ-2500 is Epson's top-of-the range conventional dot matrix printer, and it has been for a few years now. It is built like the Forth Bridge, and has by far the easiest manual control of all operational features of any printer I know. Macros are excellent, and allow an automated switch between four completely different set-up definitions in literally a couple of seconds.

Of course the recent arrival of low-cost laser printers must have eroded SQ sales, but there are still plenty of jobs that laser printers don't do very well or at all, (like printing A3 pages sideways), and their running costs are also higher than the SQ's. One laser printer limitation however is shared by the SQ-2500. Because it's not an impact device, multi-part sets are impossible, though with its speed you'd probably just print extra copies as needed.

In its target market, I doubt that the SQ-2500 has any competition. Quality is beyond question in construction, operation and output. It must be the fastest, quietest way of getting high quality print onto paper, with all the flexibility of a conventional printer and a virtual absence of operator skill or training.

The recommended retail price is £1,349. Options include a tractor unit at £90 and a double-bin automatic sheet feeder at £370. (£1,551.35, £103.50 and £425.50 including VAT.) Ink cartridges are about £24, and have a shelf life of two years.

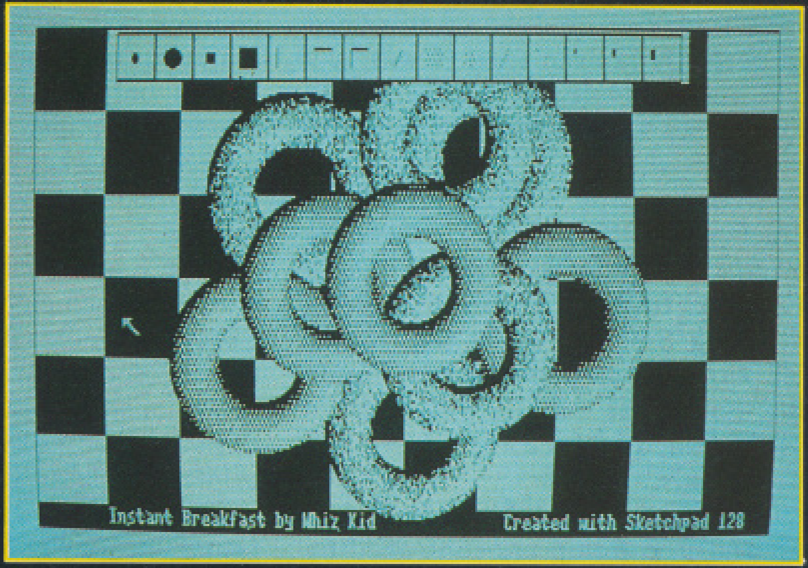
Checking current advertisements I found that the SQ-2500 can be purchased for around £975 plus VAT, with proportional reductions in the optional fittings.

TABULATION FOR YOUR COMMODORE/MAY/PRINTERS:::

	Dimensions WxDxH	Time/Speed to print 5000 chars Draft	Letter	Weight
Star LC24-10	17″x13″x4.5″	56 sec/89 cps	137secs/36 cps	14.5 Lbs
Epson LQ-500	16.5″x12″x6″	52 sec/94 cps	125secs/40 cps	15.5 Lbs
Citizen HQP=45	24″x5″x4.75″	45 secs/111 cps	100 secs/50 cps	16.5 Lbs
Epson SQ-2500	23.5x15x6	32 secs/156 cps	56 secs/89 cps	25 Lbs

Sketchpad 128





Gordon Davis examines a rare specimen – an 80- column graphics package for C128 owners

Sketchpad 128 is a rare animal. Almost all Graphics packages for the C128D operate in 40-column high-resolution mode, but this one is different – it offers a full graphics capability in 80-column mode – a resolution of 640 pixels horizontally.

The package has been made possible by the release of Walrusoft's Basic 8, a language package which we reviewed last year. Basic 8 is designed as a language which offers a graphics-orientated environment to the system designer. It's no surprise, therefore, that the salient features of Sketchpad are exactly those which Basic 8 supports most readily.

A Commodore 1351 mouse is a necessity for Sketchpad. This, I think, is unfortunate, since it isn't the sort of item that Commodore owners are likely to just have hanging around. Although, as the authors point out, a mouse is by far the best device for doing graphics, it would have cost little to allow a joystick to be interfaced with the program, just to get people started.

Little messing

Fortunately, we managed to find the office's 1351 mouse, so that play could begin. There's very little messing around following loading. Sketchpad moves straight into the drawing screen following the obligatory Basic 8 loading screen.

Free Spirit, which, as you might guess, is a Californian software house, has obviously done its Apple Macintosh homework. The main screen looks very Mac-like. All menu selections are made from the drawing screen by clicking the left mouse button. The right button is reasonably consistent in returning the user to the main menu.

The main elements on the main menu are exactly those which have been made familiar by that seminal program, MacPaint. Naturally, you can draw freehand on the screen with a variety of different pens. The indispensable airbrush is also there. The annoying thing about this is that the airbrush area cannot be sized, which is something that you can see even with C64 packages.

Sketchpad supports a suitably fast fill. This is a little worrying in the way it operates, since it seems to leave many lines unfilled along the way. But never fear, the filling algorithm gets back and fills them all back in too.

A multi-font option for writing is also available. Different fonts can be loaded from disk, and can be typed anywhere on the screen in one of five different sizes. A useful touch is that Return will align the text cursor exactly one line below the text you've already put in.

This is not a multi-colour graphics program, quite rightly, because unless you are designing loading screens or demos, anything other than two-colour screens is functionally useless

for all owners of black-and-white printers. Even so, you can alter the foreground/background to give you more screen readability.

Although clearing the screen is fairly deadly on Sketchpad – there is no check on this, you can lock your program into memory, which will put it into a spare area of Ram for safety. This is also useful if you're uncertain about the changes you're making to a picture.

The graphics shapes options are the ones that show the most difference over most graphics packages. All the traditional favourites are there: lines, boxes, circles, and so on. Producing these is more or less as straightforward as the commands on any other graphics package. A unique extra, however, is the addition of the so-called Rylander shapes.

Solid graphics

Rylander shapes are views of shaded solids of three types: cylinder, sphere and torus. They are named after the programmer who designed a set of algorithms for representing them easily.

Basic 8 later incorporated these calculations, and here they are at last in a graphics package. The shading on the Rylander shapes can be varied between pixel dots or zonal shading, and whatever you use them for, they are an interesting addition.

Besides these shapes, another useful feature is the inclusion of arcs. You can provide arcs of any pitch and angle. Rays can also be provided, subtended at any angle from a central point. Both of these make the production of piecharts an easy matter.

One vital thing I spent ages looking for was some form of Zoom option. It's very uncertain using a graphics package, and some means of adjusting things on a pixel-by-pixel basis can be useful. This is, annoyingly, not available from the main screen, but from a sub-menu called the Gadgets menu.

Besides a pixel zoom, Gadgets also supports a Cut/Paste option which allows you to move chunks of the picture around, or simply remove them. You can also scroll your picture – an interesting feature which few packages support. This menu also supports a number of patterns which can be used to fill areas. Unfortunately, you have to drop the clip patterns in yourself – there is no automatic fill for these.

Free Spirit deserves complimenting for the number of features which have been included in Sketchpad – I haven't covered everything here. The main flaw in the program is that it is a trifle slow, due to being written entirely in Basic 8.

I'm also not sure about some of the design priorities – some of the essentials are very hard to get to. The program documentation is brief and unhelpful. These days, this is something I expect software houses to get right. On the other hand, C128 80-column drawing packages are fairly hard to come by, so it seems unfair to cavil too much.

Touchline:
Title: *Sketchpad 128.* ***Supplier:*** *FSSL/Free Spirit, 18 High St., Pershore, Worcestershire, WR10 1BG.* ***Tel:*** *(0386) 553153.* ***Price:*** *£24.95.*

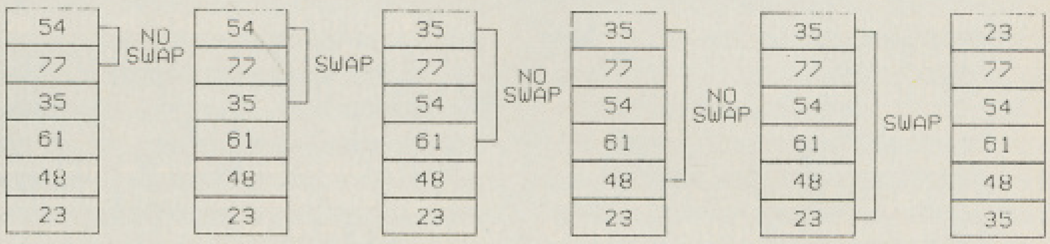
Routine Programming

A bubbly routine to sort lists into orderly sequences

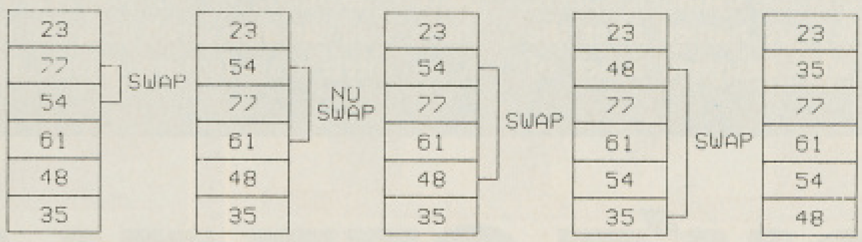
By Eric Doyle

Programs often include lists of numbers or strings which have to be displayed in an orderly way. This could be alphabetically or numerically. This bubble sort subroutine can be used, suitably modified, for either purpose.

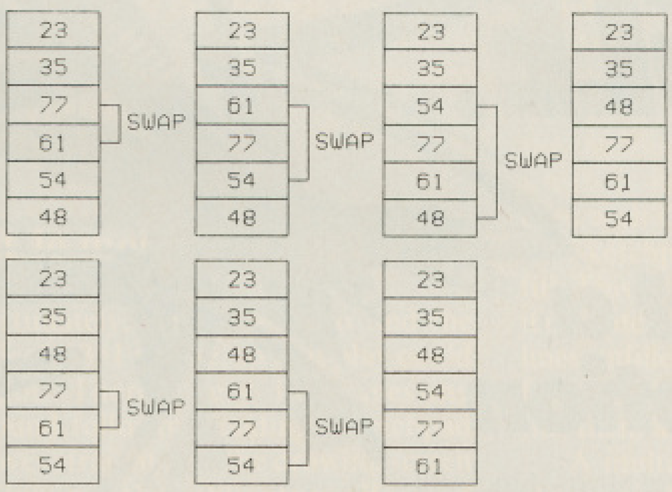
Bubble sorts work by comparing neighbouring list items and swapping them over if one exceeds the other. Take this list as an example:



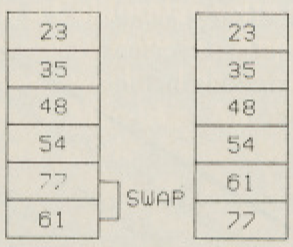
The first number is compared to the rest of the numbers in the list one at a time. If the number under comparison is smaller than the first number, a straight swap takes place. In this way the smallest value rises to the top.



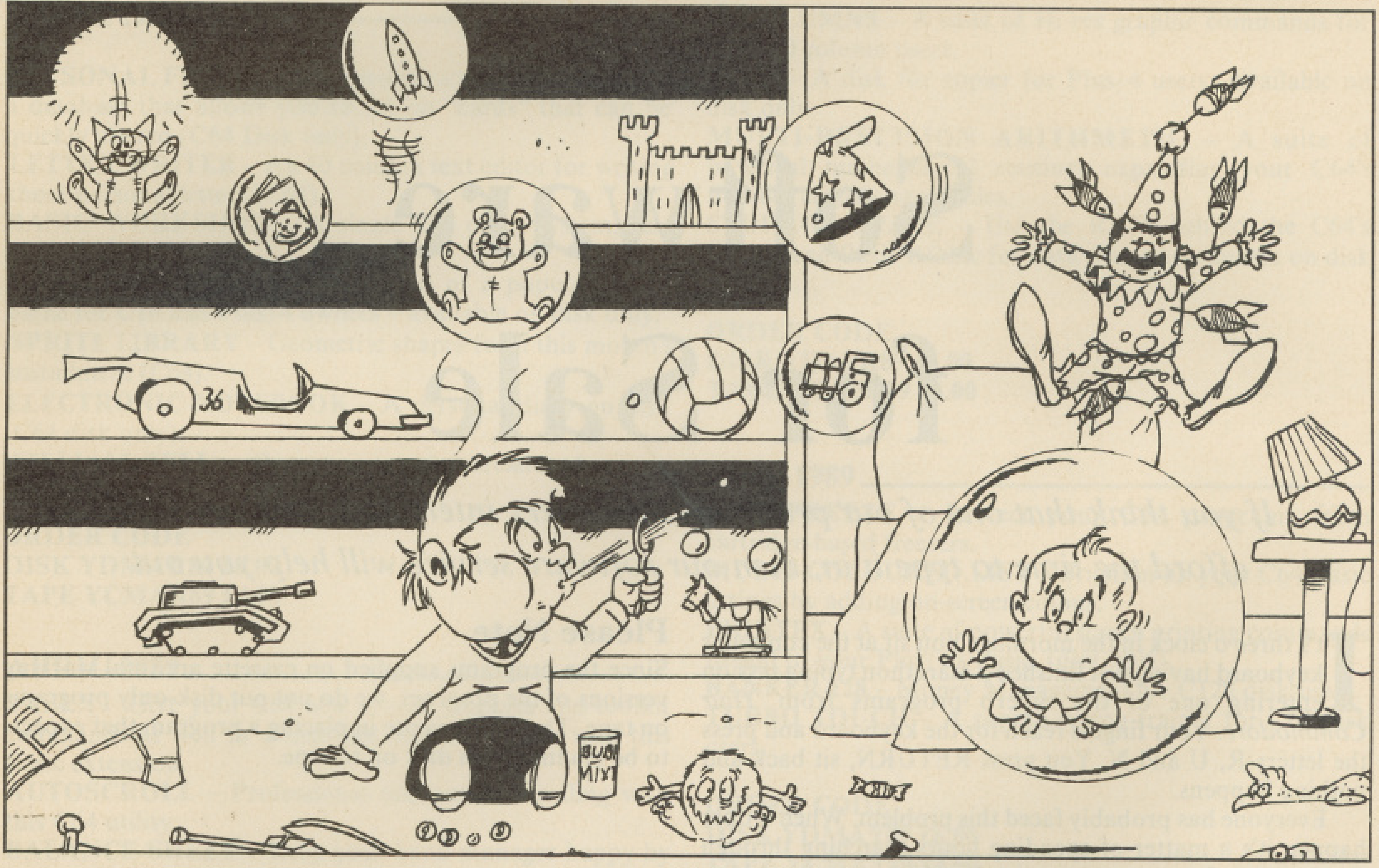
After this process is completed, the second number is compared to all of the following figures in a similar way.



This is repeated, gradually passing down the column until the last two characters are compared and swapped if necessary, and the sort is then complete.



The two elements of the sort are a loop which selects the character to be tested, and a loop within this, a nested loop, which selects the remaining characters in the list for comparison.



```
60200 FORZ1=1TOZ-1
60210 Z2=Z1+1
60220 FORZ3=Z2TOZ
```

The value Z is the number of items in the list. Line 60200 forms the main loop for the character to be tested. It only runs from the first to the master penultimate character because there would be no point in comparing the final character with itself.

Line 60210 calculates the first character to be tested by the nested loop which is initiated in line 60220. Line 60230 tests the two values, and if they are already in the correct order no action is taken and the exchange routine is bypassed.

```
60230 IFZ(Z1)<Z(Z3)THEN60270
```

If an exchange is necessary, this is done by using a temporary store for one of the values, so that values can be swapped with ease.

```
60240 Z(0)=Z(Z1)
60250 Z(Z1)=Z(Z3)
60260 Z(Z3)=Z(0)
```

Z(0) takes the value held in Z(Z1) and Z(Z1) can then be given the value held in Z(Z3). The stored value can then be moved from Z(0) to Z(Z3), and Z(0) may then be discarded.

Now that everything has been done, the loops can be closed and eventually control is handed back to the program with a RETURN command.

```
60270 NEXTZ3,Z1
60280 RETURN
```

The listings shown contain an example of numerical sorting using this subroutine and there is also an alphabetical sort routine which shows how this routine has to be modified for this use. Really all that has happened is that the Z(x) values are changed to Z$(x).

Parameters for Main Program

Sending:

Z Number of list items
Z(x) List item values

Returns:

Z(x) The sorted list

Other Variables Used

Z1 Main loop variable
Z2 Remainder of list items
Z3 Nested loop variable

```
PROGRAM: NUMBER SORT

33  10 A=1:AA=50:DIMZ(AA),ZZ(AA)
09  20 R%=RND(1)*AA+1
E9  30 Z(A)=R%:A=A+1:IFA<AA+1THEN20
54  40 FORA=1TOAA:PRINTZ(A);:NEXT:PRINT
BE  50 Z=A-1:PRINT"SORTING..."
32  60 GOSUB1000
DF  70 FORA=1TOAA:PRINTZ(A);:NEXT
D0  80 END
3A  1000 FORZ1=1TOZ-1
77  1010 Z2=Z1+1
B7  1020 FORZ3=Z2TOZ
75  1030 IFZ(Z1)<Z(Z3)THEN1070
71  1040 Z(0)=Z(Z1)
22  1050 Z(Z1)=Z(Z3)
F7  1060 Z(Z3)=Z(0)
70  1070 NEXTZ3,Z1
B2  1080 RETURN
```

```
PROGRAM: NAME SORT

53  1 Z$(1)="HENRY"
17  2 Z$(2)="JEREMY"
94  3 Z$(3)="JOHN"
AF  4 Z$(4)="ALBERT"
18  5 Z$(5)="JON"
B0  6 Z$(6)="WILLIAM"
2C  7 Z$(7)="STUART"
EE  8 Z$(8)="CHRIS"
50  9 Z$(9)="CHRISTOPHER"
02  40 FORA=1TO9:PRINTZ$(A)" ";:NEXT:PRINT:PRINT
37  50 Z=A-1
32  60 GOSUB1000
98  70 FORA=1TO9:PRINTZ$(A)" ";:NEXT
D0  80 END
3A  1000 FORZ1=1TOZ-1
77  1010 Z2=Z1+1
B7  1020 FORZ3=Z2TOZ
61  1030 IFZ$(Z1)<Z$(Z3)THEN1070
6D  1040 Z$(0)=Z$(Z1)
A0  1050 Z$(Z1)=Z$(Z3)
E5  1060 Z$(Z3)=Z$(0)
70  1070 NEXTZ3,Z1
B2  1080 RETURN
```

Software for Sale

If you think that one of our programs looks very interesting, but you can't afford the time to type it in, then our software service will help you out

It's three o'clock in the morning. You sit at the computer keyboard having just finished a marathon typing session entering one of the superb programs from *Your Commodore*. Your fingers reach for the keyboard and press the letters R, U and N. You press RETURN, sit back and nothing happens.

...Everyone has probably faced this problem. When it does happen it's a matter of spending hours searching through the program for any typing mistakes. No matter how long you look or how many people help you, you can usually guarantee that at least one little but slips through unnoticed.

The *Your Commodore* Software Service makes available all of the programs from each issue on both cassette and disk at a price of £6.00 for disk and £4.00 for cassette. None of the documentation for the programs is supplied with the software since it is all available in the relevant magazine. Should you not have the magazine then back issues are available from the following address:

INFONET LTD, 5 River Park Estate, Berkahmsted, Herts, HP4 1HL.
Tel: (04427) 76661

Please contact this address for prices and availability.

The Disk

Programs on the disk will also be supplied as totally working versions, i.e. when possible we will not use Basic Loaders thus making use of the programs much easier. Unfortunately at the moment we cannot duplicate C16 and Plus/4 cassettes. However programs for these machines will be available on the disk.

What programs are available?

At the top of each article you will find a strap containing the article type, C64 Program etc. So that you can see which programs are available on which format, you will also find a couple of symbols after this strap. The symbols have the following meaning:

This symbol means that the program is available on cassette.

These programs are available on disk.

Please Note

Since the programs supplied on cassette are total working versions of the program, we do not put disk-only programs on tape. There is no sense in placing a program that expects to be reading from disk on to tape.

JANUARY 1989

PREFAB SPRITES – A powerful sprite editor for the C64.
DAZPRO – A simple but helpful text processor for the C64. Available on disk and cassette but will only store files on tape.
UDG COMPRESSOR – Save on memory when using UDG's in your programs. For C64 only.
WILLIAM TELL – Our popular arcade game for the C64.
+4 AUTORUN – Improve tape loading on your Plus/4 cassette. Only available on disk.
MINIBASE – A database for C128 owners.

ORDER CODE
DISK YDJAN89 £6.00
TAPE YCJAN89 £4.00

FEBRUARY 1989

TAPE MENU – Add a menu system to your program cassettes (C64).
SONIC EFFECTS – A superb sound editor for the C64.
F DUMP – Dump your C64 text screens to printer with ease.
DATA LOADER – A simple way to enter those reams of C64 DATA lines.
SPRITE LIBRARY – A collection of birds to your growing library (C64).
PLAY THE GAME – A superb fruit machine programe for the Plus/4. (Available on disk only).

ORDER CODE
DISK YDFEB89 £6.00
TAPE YCFEB89 £4.00

MARCH 1989

PERSONAL FILE – A cross between a wordprocessor and a database that allows you to set up "cards" that can be quickly altered (C64 Disk only).
LETTER WRITER – An 80 column text editor for writing those personal letters (C64).
BASIC WORKSHOP – A single key entry system, just like a Speccy! (C64)
HEAD FOR HOME – Our version of a popular board game for C16 and Plus/4 owners – available on disk only.
SPRITE LIBRARY – Geometric shapes form this month's installment (C64).
ELECTRONIC NOTEBOOK – A personal diary on disk (C64 disk only).
WILLIAM TELL – Our very popular arcade adventure for the C64.

ORDER CODE
DISK YDMAR89 £6.00
TAPE YCMAR89 £4.00

APRIL 1989

BASEX – Give your C64 new sound, graphics and toolkit commands as well as a machine code assembler with this Basic extension.
AUTOSCROLL – Professional text screen scrolling with this C64 utility.
BALANCE SHEET – Keep your bank manager happy by keeping better track of your money with this C64 program (disk only).
ICON 64 – Add icon control to your own programs with this C64 utility.
80 COLUMNS – A suite of Hi-res graphic commands for C128 80 column users.
ZEUS – A disk file copier for Plus/4 users. Available on disk only.
MULTI-PRECISION ARITHMETIC – A suite of powerful mathematical routines expanding your C64's number crunching abilities.
STORAGE SPACE – Use the RAM behind the C64's Kernal and Basic ROMs for data storage (available on disk only).

ORDER CODE
DISK YDAPR89 £6.00
TAPE YCAPR89 £4.00

MAY 1989

ANTI-FREEZE – Protect your C64 programs from cartridge-based freezers.
MULTI-COLOURED LIST – Brighten up your C64 Basic listings by adding on-screen colour.
PRINTFX – A suite of powerful screen printing commands to add to your Plus/4's Basic. (disk only).
BANKER 128 – Money Management for C128 disk users.
TURBO TOUCH – A superb typing trainer for C64 disk users.

ORDER CODE
DISK YDMAY89 £6.00
TAPE YCMAY89 £4.00

Cassettes or disks are available from March 1986. Please ring the editorial office (01-437 0626) for details of these.

ORDER FORM – PLEASE COMPLETE IN BLOCK CAPITALS

NAME	QTY	TAPE/DISK	ORDER CODE	PRICE
JUNE '89		TAPE (£4.00)	YCJUNE '89	
JUNE '89		DISK (£6.00)	YDJUNE'89	
OVERSEAS POST £1				
			TOTAL	

NAME...

ADDRESS...

POSTCODE...

I enclose a cheque/postal order for £....... made payable to ARGUS SPECIALIST PUBLICATIONS LTD.
All orders should be sent to: *YOUR COMMODORE,*
READERS SERVICES, ARGUS HOUSE, BOUNDARY WAY, HEMEL HEMPSTEAD, HERTS. HP2 7ST

Please allow 28 days for delivery.

YOUR COMMODORE

Lineage: 58p per word.
(including VAT)
Semi display: £10.95 plus VAT per single column centimetre minimum 2cm. Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

0442 66650

Send your requirements to:
CLASSIFIED DEPARTMENT
ASP LTD. ARGUS HOUSE,
BOUNDARY WAY,
HEMEL HEMPSTEAD HP2 7ST.

SOFTWARE

SAMDALE COMPUTERS LTD
"Better than the best of the rest"
RING NOW Credit Card Hotline 0621 742617
THE LATEST SOFTWARE FOR COMMODORE C64/C128/AMIGA
Ring now for a free price list on 0621 742617.
or send stating name and address and machine type.
SAMDALE COMPUTERS LTD
1a Austral Way, Highfield Rise, Althorne, Essex CM3 6DN.

SERVICES

COMMODORE SUPPLIES

C64 Power Supply	**£19.95**
CBM Plus 4 Power Supply	**£29.95**
C-16 Power Supply	**£12.95**
1530 Datasette	**£27.95**
1531 datasette	**£27.95**
Azimuth head alignment kit	**£9.95**
Serial Lead (for d/drive/printer)	**£5.95**
C64 reset Switch	**£4.95**
Dust Cover (please state computer)	**£3.50**

Prices include VAT & P&P. Allow up to 14 days for delivery. Chq/PO to:
Omnidale Ltd, 23 Curzon Street, Derby, DE1 2ES. Tel 0332 291219.

HARDWARE

COTSWOLD COMPUTERS
MILTON KEYNES
Call today for the best prices in the Area.
AMIGA A500 inc MODULATOR
£374
AMIGA A500 + 10 Star Games Pack
£389
ST EXPLORER PACK
£299
Plus all software at discount prices
All prices include V.A.T.
22 Polrlian Place, Fisher Mead, Milton Keynes MK6 2LH
Phone (0903) 662331

SOFTWARE

YOUR COMMODORE SPECIALS

TECH DRAW 64 - A comprehensive technical illustration aid for C64 owners for both disk and tape. Commands available include LINE DRAW, RAYS, BOX, CIRCLE, ARC, ELLIPSE, FREEHAND, FILL, HATCH & TEXT. Wipe copy, move and rotate, save and load or print your drawing all for £8.95 tape or disk. Order Codes TD64T (tape) TD64D (disk)
Full details in Your Commodore July 1987.

SPEEDY ASSEMBLER
Your Commodore's very own Assembler, a 100% memory resident program for loading from tape or disk, featured in the Your Commodore Machine Code Series and now the Y.C. standard assembler only £9.95 Tape or Disk. Order Codes YSPASSD (disk) YSPASSR (tape)
Order from Your Commodore Readers Services at 9, Hall Road, Maylands Wood Estate, Hemel, Hempstead Herts HP2 7BH. Please make cheques payable to A.S.P. Ltd., or telephone your Access/Visa order on 0442 41221

SPECIAL OFFER

T.C. COMPUTERS
Tel: Wantage 02357 60177
AMIGA A500 + Software from **£349**
AMIGA A500 + 1084S Stereo Monitor + Software**£599**
AMIGA A500 + 10 Star Pack **£379**
AMIGA AIR MILES PACK **PHONE**
AMIGA System 500 Colour **£749**
Commodore PC1 Mono +Software **£339**
Also Atari ST amd PC Compatibles including Amstrad available.
ENQUIRIES WELCOME
Please send S.A.E. for details:-
NOW AT:-
12A Barnards Way, Charlton Heights, Wantage, Oxon OX12 7EB

You'll never believe OUR PRICES

3.5" Disks x 10 only	**£9.99**
5.25" disks x 10 only	**£4.50**
Storage Box 25 x 3.5"	**£3.99**
Storage Box 50 x 3.5"	**£5.95**
Storage Box 80 x 3.5"	**£7.99**
Storage Box 100 x 5.25"	**£7.99**

ALL PRICES INC VAT & CARRIAGE
Crofton Micro Supplies
FREEPOPST
London. SE4 2BR
Telephone 01-469-3246

THE COPY DEADLINE FOR THE AUGUST ISSUE IS 31st MAY. SERIES DISCOUNTS ARE AVAILABLE ON SEMI-DISPLAY BOOKINGS.

WHY NOT ADVERTISE YOUR PRODUCTS AND SERVICES IN YOUR COMMODORE
CHARLES UCHEOBI
0442 66650

FOR SALE

esoteric
DISK MATE 1541/C64 Disk Toolkit
SECUREWORD WP Data Encryption
PSYCHOLOGY + Analyse your character.
Disk Only. **£8.00** inc p&p
15 Lanmoor Est. Lanner, Redruth, Cornwall. TR16 6HN

CLASSIFIED COUPON

ALL CLASSIFIED ADVERTISEMENTS MUST BE PRE-PAID. THERE ARE NO REIMBURSEMENTS FOR CANCELLATIONS.

I enclose my Cheque/Postal Order for £ ....for .... insertions, made payable to Argus Specialist Publications.
(Delete as necessary)
PLEASE DEBIT MY ACCESS/BARCLAYCARD NO

EXPIRY DATE ....

£...............FOR ....INSERTIONS
Name..

..........................POST CODE ....

DAYTIME TEL NO ....

Signature ....Date ....

CLASSIFIED DEPT. ARGUS HOUSE, BOUNDARY WAY, HEMEL HEMPSTEAD HP2 7ST.
RATES: Lineage 58p per word (VAT inc) Semi-display: £10.95 + VAT per single column cm minimum size 2cm.
Series discounts available.

☐ **FOR SALE** ☐ **SOFTWARE** ☐ **SPECIAL OFFERS** ☐ **OTHER STATE**

Repairs Guide

0442 66650

CROYDON COMPUTER CENTRE

25 Brigstock Road, Thornton Heath, Surrey, CR4 7JJ Tel: 01 683 2646

COMPUTER SERVICING

(Est'd since 1979)

We repair — on the premises — quick turnround

- ★ Commodore
- ★ BBC & Electron (Approved Service Centre)
- ★ Amstrad & Torch (Approved Service Centre)
- ★ Disc Drives, Printers, Monitors

Mail Orders by phone Access & Visa accepted

COMMODORE REPAIRS

(with FREE extended guarantee)

COMMODORE 64	£28.70
COMMODORE 128	£36.70
COMMODORE C16/+4/VIC 20	£20.70

Price is all inclusive and we guarantee the whole computer for six months

KPR MICRO,
34, Petunia Crescent, Springfield,
Chelmsford, Essex CM1 5YP
Tel: 0245 - 468554

COMMODORE SPARES & REPAIRS

REPAIRS OFFERS.

C64, C+4 .... £20 inc VAT +PP
1541 .... £28 inc VAT + PP

Three month warranty, 1 week turnround

SPARES

90614	£9.30	8501	£8.95
6510	£9.95	C64 Power Pack	£22.00
6526	£9.50	C64 Cassette Unit	£22.00
6581	£11.25	C128 Power Pack	£69.00

Inc VAT & PP

OASIS COMPUTER SERVICES
Dept 4C, 14 Ridgeway Road, Salisbury, Wiltshire, SP13BU. Tel (0722) 335061 Ext 214

"COMPUTER KAPUT?"

"BETTER GET" Readidata THE COMMODORE SPECIALIST Private & Trade Service

PET V1C20
PLUS4 C16
C64 C128
AMIGA...

FAST EFFICIENT SERVICE

FOR REPAIRS CALL 01-861 2835

NW LONDON-HARROW

VTS COMPUTER REPAIRS

01 - 407 9829

Simply send your unit to the address below securely packed with fault description and max fee.
COMMODORE, AMSTRAD, BBC, SINCLAIR + MORE.........£28.00

VTS COMPUTER REPAIRS

12 Lyly House, Gt. Dover Street, London, SE1 4EQ.
FAST, RELIABLE TURNAROUND

PROBLEMS WITH YOUR COMMODORE ?

FOR FAST, RELIABLE AND PROFESSIONAL REPAIRS AT COMPETITIVE PRICES

P M ENGINEERING

UNIT 260, MILTON SCIENCE PARK, CAMBRIDGE CB4 4WE

VISA

ST. IVES (0223) 420007

We can also supply Hardware, Software, Blank Disks, Accessories and Spare parts.
C64 Repairs £25.00 inc. VAT (p&p FREE)

REPAIRS

AMIGA ATARI etc.
MONITORS & ALL ADD ON,s.
AMIGA UPGRADES AND MOD,s
All estimates FREE
Prices from only £19.50
Call or post *FAST TURNAROUND*

PVS ELECTRONIC COMPONENTS
244 Deansgate Court,
Deansgate, Manchester M3 4BQ
Tel: 061 831 7086 Telex 668986
Fax 061 832 6934

PVS

COMPUTER REPAIRS

Fixed Super Low Prices!
Inclusive of parts, labour and VAT

FIRST AID FOR TECHNOLOGY

1 week turnaround Collection/delivery available for local area

★ SUPER OFFERS ★

SPECTRUM	£14 inc. + Free Game	ELECTRON	£32
C64	£22 inc. + Free Game	COMMODORE 1541 DISC DRIVE	£38 inc.
C16	£18 inc.		
VIC20, C+4	£22inc.	SPECTRUM 128	£16 inc.
BBC	£32 inc.	C64 P.S.U. for sale	£20 inc.

Please enclose payment with item - 3 month warranty on repair. Please enclose advert with repair

W.T.S. ELECTRONICS
Y.C. Studio Master House, Chaulend Lane, Luton, LU4 8EZ. Tel: 0582 491949

HINDLEY ELECTRONICS

The Computer repair specialists

SAME DAY SERVICE (CALLERS)

FAST

24hr TURNAROUND (POSTAL)

EXPERT COMPUTER REPAIRS

COMMODORE — SINCLAIR — AMSTRAD — ACORN — BBC
STANDARD CHARGES (INC VAT AND FULLY INSURED RETURN POSTAGE)

Commodore 64	£25.00	Commodore 64C	£29.00	Commodore 128	£32.00
Commodore +4	£25.00	Commodore vic 20	£18.00	AMIGA 500	£35.00
1541 disc drive	£32.50	(OTHER DRIVES REPAIRED RING FOR DETAILS			
Spectrum 48k	£18.00	Spectrum + 2	£25.00	Amstrad cpc 464	£25.00
		Acorn Electron	£21.00		

3 MONTHS "NO QUIBBLE" WARRANTY ON ALL REPAIRS
SPARES — SPARES — SPARES — SPARES
Low prices E.G Spectrum 48k MEMBRANE £3.99 Spectrum + £6.99 LOW PRICES
COMPUTER ICS and GENERAL COMPONENTS WHY PAY MORE (Send large SAE for full lists)

HINDLEY ELECTRONICS
DEPT A 97 MARKET STREET, HINDLEY WIGAN WN2 3AA Tel (0942) 522743

Computer repairs: Callers requiring same day service please ring before calling and add 10% to standard charge - Standard charges do not include repairs to external items E.G power units tape recorders joysticks etc. A surcharge may be levied for machines that have sustained severe damage E.G fire flood tamper

B R E A K

Readers Problems

Though the Commodore 64 is one of the world's most popular microcomputers, it can be very difficult to find specific information about your particular machine.

At the *Your Commodore* office we receive literally hundreds of letters from you, our readers, on a wide range of subjects ranging from the simple 'Can you give me the telephone number for...', to the more complex 'I'm trying to write a program that uses a split screen. How do I do it?'

Unfortunately, the volume of mail received has become so great that it is impossible to answer every letter and still manage to publish a magazine each month.

For this reason we have felt it necessary to produce a number of guidelines for getting information from us:

1) We cannot guarantee to answer every letter sent to the magazine. Should it become apparent that a number of readers are suffering from the same problem, then we will reply to the letter via the Letters page.
2) A new helpline has been set up. This will be open for your queries on Tuesday and Thursday afternoons between 2.00pm and 4.00pm. We will not be able to deal with our telephone queries at any other time. If our technical adviser is not available when you ring, then a message will be taken.
3) If you are having problems with one of our listings, can you please let us know in writing. This will enable us to see if a number of people are having the same problem. When a common problem becomes apparent with a program, then a correction sheet will be issued. Enclose a self-addressed, stamped envelope and we will send you a copy of the correction sheet as soon as it is available.

We are sorry that it has become necessary to instigate these rules. However, we are sure that you will agree with us that the more time that we can spend making *Your Commodore* the most informative magazine around, the better.

For program queries write to:
Program Corrections
Your Commodore
Argus House,
Boundary Way,
Hemel Hempstead
HP2 7ST
Tel: 0442-66551

CORRECTIONS FOR MARCH '89

HEAD FOR HOME

Once again those indestructible gremlins have caused havoc.

The program as it stands will not run correctly. This is due to the fact that quite a bit of code is missing from the loader program. An amendment is on its way. The update will be published as soon as it reaches our office. We apologise for any inconvenience to our Plus 4 readers.

CORRECTIONS FOR APRIL '89

BALANCE SHEET

Unfortunately there appeared a couple of errors in the Balance Sheet program from the April Edition.

1) Page 331, right hand column, paragraph 4 should read Load "LIST4" and not Load "LIST1".
2) The second line of Listing 1 should be POKE770,131:POKE771,164
3) All the REM statements should be taken out of Listing 1.
4) Again on page 31, right hand column, paragraph 1. This should read:
50 SYS57812"@:SHEET",8,1

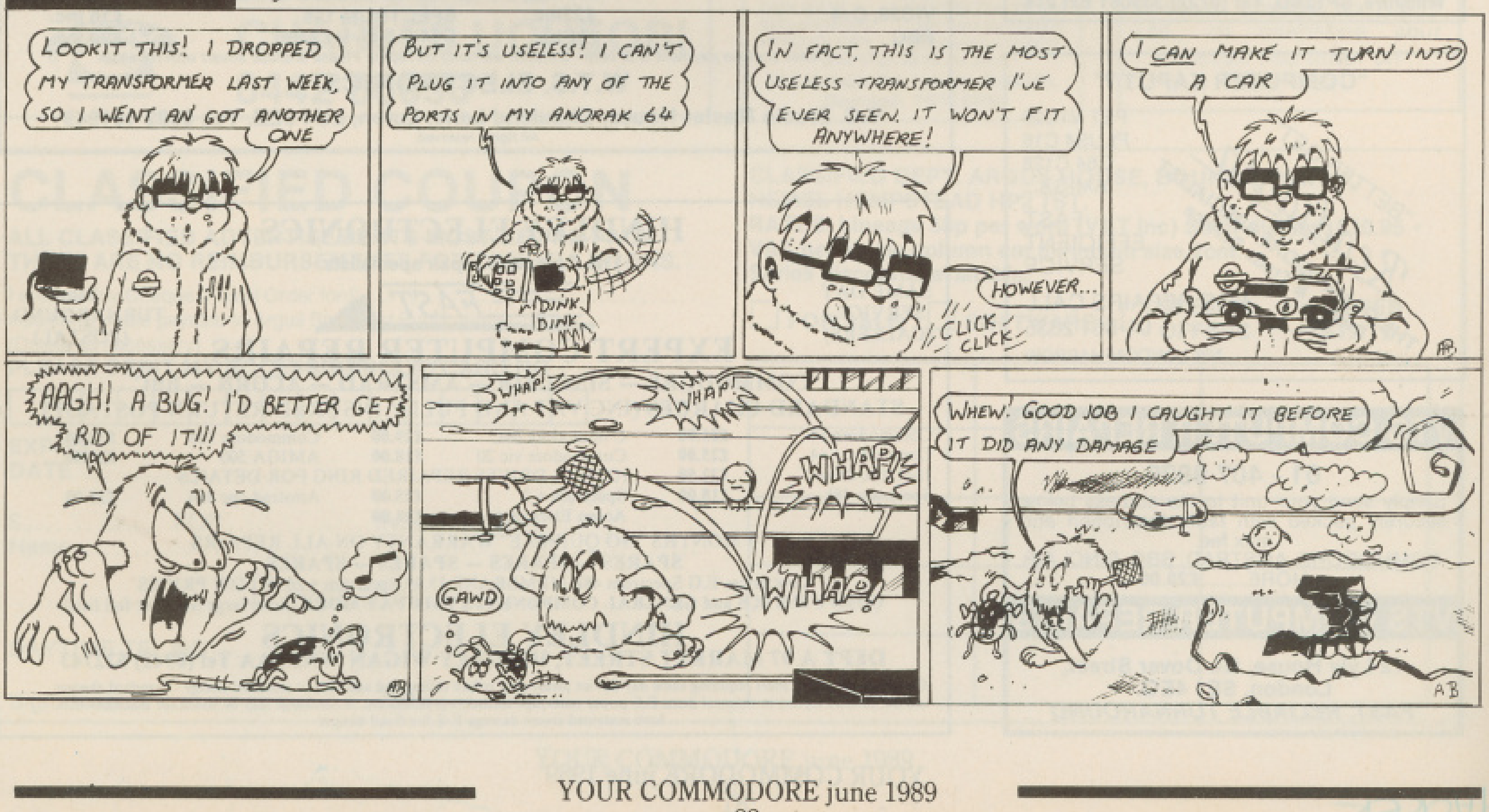
Commodore Where Are You?

At the *Your Commodore* office we are repeatedly asked for the address and telephone number of Commodore U.K. Many people, after referring to their computer manuals, believe them to be based in Corby.

The Commodore plant at Corby was closed down some time ago. Reproduced here you will find the correct address for Commodore U.K.

Commodore Business Machine, (UK),
Commodore House, The Switchback,
Berks SL6 7XA.
Tel: (0628) 770088
Gardner Road, Maidenhead,

The Nibbles *By Alan Batchelor*





THE TIME HAS COME . . .

Britain's First Music Magazine for the Computer User!
Featuring Reviews on Hardware & Software across all formats – Micro Music is the magazine the Computer Musician has been waiting for.
Available 10th March from all good Newsagents


Emlyn Hughes
INTERNATIONAL
SOCCER
GAME OF THE YEAR!
WINNER OF CCI OSKARS FOR BEST SPORTS SIMULATION AND BEST GAME OF THE YEAR!
Quite simply the best football simulation available for the 64 Nothing short of superb.
ZZAP 64
AVAILABLE NOW FOR COMMODORE 64
£9.95 tape £12.95 disk
Coming soon for Spectrum & Amstrad
Audiogenic
Audiogenic Software Limited, Winchester House, Canning Road, Harrow HA3 7SJ, England
Order by phone on 01 861 1166
VISA